Deterministic Escape from Local Minima: Oracle from Simulated Over-parametrization [*]

Tianqi Shen
Department of Computer Science
City University of Hong Kong
Hong Kong
tianqshen5-c@my.cityu.edu.hk

Kunhan Gao
Department of Data Science
City University of Hong Kong
Hong Kong
kunhangao2-c@my.cityu.edu.hk

Ziye Ma
Department of Computer Science
City University of Hong Kong
Hong Kong
ziyema@cityu.edu.hk

Abstract

Modern machine learning problems are predominantly non-convex, often containing a potentially infinite number of local minima that can hinder gradient-based algorithms from converging to global solutions or those that generalize well. Since traditional wisdom usually assumes that local solutions are not escapable via such algorithms, existing works mostly focused on how randomized updates could escape saddle points to combat the complex landscape. In this work, we challenge this belief by demonstrating a deterministic approach for escaping local minima in the non-convex matrix sensing problem. We introduce LEAC (Lift, Escape, Approximate, and Collapse), a new algorithm that leverages simulated over-parametrization to find deterministic escape directions. Starting from a local minimum in the matrix space, LEAC exploits tensor-space geometry to approximate the effect of a high-dimensional escape step. This deterministic update guarantees a strictly lower objective value than that of the original local minimum. Our theoretical analysis establishes formal conditions for successful escape, and empirical results show that LEAC achieves superior optimization performance at a fraction of the computational cost of fully over-parameterized methods. By demonstrating that deterministic escapes are achievable, our work pioneers a new line of research aimed at escaping local minima with first-order optimizers, and lays the groundwork for more general optimization algorithms endowed with similar capabilities.

1 Introduction

The challenge of optimization in modern machine learning landscapes, all inherently non-convex, cannot be overstated. The inherent difficulty arises because non-convex problems can encode NP-hard problems, such as the subset sum problem, making them at least NP-hard in the general case [1]. Moreover, given the enormous size and complicated structures of modern neural networks, their optimization landscapes are filled with multiple local minima, saddle points, and possibly even flat regions. Unlike convex problems, where any local minimum is also a global minimum, the lack of such guarantees is a major obstacle to efficient training [2]. These landscapes can deceive local optimization methods, which might converge to a suboptimal point or fail to converge at all [3].

Traditionally, it has been assumed that local minima are inescapable black-holes for gradient-based methods, leading to a significant body of research focused on dealing with saddle points instead. One of the most common approach is to use randomized updates to escape saddle points with high probability, pioneered by [4]. From a dynamical system's standpoint, local minima are stable equilibria of the system, attracting all solution trajectories with a sufficiently small step-size. Therefore, learning rate adjustment [5, 6, 7] emerges as a possible way to escape, while none of the existing works could provide guaranteed escape beyond random occurences.

In this work, we aim to illuminate the potential of deterministic algorithms for conquering the non-convex landscape. Specifically, we demonstrate that gradient methods can escape not only saddle points but also local solutions and spurious second-order points in matrix sensing problems. Matrix sensing, which is crucial in various signal processing and machine learning applications, involves recovering a low-rank matrix from a limited set of linear measurements.

[*]Preprint.

This inherently non-convex problem often leads to local minima, where traditional first-order optimizers tend to fail [2]. The objective of matrix sensing is to recover a low-rank matrix M^* from linear measurements of the form $b = \mathcal{A}(M^*)$, where $\mathcal{A}$ is a linear function, with details deferred until later sections.

Previous studies have revealed a key insight in solving this non-convex challenge, which is that over-parameterization, a technique where the model is made larger than strictly necessary, can substantially improve the optimization landscape of the problem [8, 9, 10]. In particular, [10] introduced a tensor-based framework that lifts the original decision variable from a matrix to higher-order tensors. This approach has the remarkable ability to transform spurious local solutions—local solutions that are not globally optimal—into strict saddle points, which possess strict descent directions, effectively reshaping the optimization landscape. This discovery leads us to ponder the question of *whether this descent direction also exists in the un-lifted matrix space and will it lead to a decrease in objective value?* Since full tensor lifting introduces computational overhead that increases exponentially with problem size, exploring the ability of over-parametrized methods to escape spurious solutions without this computational burden becomes particularly crucial.

To this end, we propose a deterministic framework that combines the efficiency of matrix space optimization with the global guarantees of tensor space optimization. The key insight of our approach is to exploit escape directions provided by tensor lifting only when local minima are encountered. Specifically, our framework operates as follows: In the first phase, optimization begins in the matrix space, where gradient descent is applied until convergence to a local minimum. If a local minimum is detected, the algorithm will enter into second phase, where a deterministic descent direction is calculated as if the local minimum was lifted into tensor spaces, which is why we call it simulated over-parametrization. This design reduces the computational burden compared to full tensor lifting while retaining its ability to escape local minima.

We validate our approach through numerical experiments on a benchmark problem, Perturbed Matrix Completion [9], which is commonly used to evaluate optimization frameworks in matrix sensing. Although the experiments are relatively simple and focus primarily on small- to medium-scale settings, they effectively demonstrate that our algorithm can deterministically and verifiably escape local minima. This capability represents a pioneering and critical advancement for non-convex optimization methods.

The remainder of the paper is organized as follows. Section 2 provides an overview of necessary backgrounds. Section 3 introduces an intermediary algorithm LEAC-Canonical to help readers gain more intuitive understandings, which further leads to our main results discussed in Section 4, with LEAC-Simulated being our new algorithm. Section 5 presents numerical experiments to demonstrate the effectiveness of our new approach.

2 Background and Related Work

Matrix sensing plays a critical role in signal processing and compressed sensing, where it facilitates the recovery of signals or images from incomplete or corrupted matrix measurements [11, 12, 13]. The objective is to reconstruct an unknown low-rank matrix $M^* \in \mathbb{R}^{n\times n}$ from a set of m linear measurements, expressed as:

$$\mathcal{A}(M^*) = \left[\langle A_1, M^*\rangle, \langle A_2, M^*\rangle, \cdots \langle A_m, M^*\rangle\right]^\top, \tag{1}$$

where $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ is a linear measurement operator defined by m symmetric sensing matrices $\{A_i \in \mathbb{R}^{n\times n}\}_{i=1}^m$, and $\langle A_i, M^*\rangle = \mathrm{Tr}(A_i^\top M^*)$ represents the matrix inner product. The goal is to recover M^* from b when $m \ll n^2$, under the assumption that M^* is low-rank. Such problems arise in various applications, including collaborative filtering [14], quantum state tomography [15], and visual tracking [16]. Furthermore, this problem is directly related to the training of certain types of neural networks [17] and can represent general polynomial optimization problems [18]. As a result, matrix sensing serves as an excellent benchmark for studying and understanding more complex machine learning challenges from a theoretical perspective.

To recover M^*, the matrix sensing problem is typically formulated as the following optimization task:

$$\begin{aligned}\min_{M\in\mathbb{R}^{n\times n}} & f(M) = \frac{1}{2}\|\mathcal{A}(M) - b\|_2^2,\\ \text{s.t. } & \mathrm{rank}(M) \le r,\end{aligned} \tag{2}$$

where r denotes the rank of the ground-truth matrix M^*. However, solving this rank-constrained problem is computationally intractable due to the non-convex nature of the rank constraint, which makes the problem NP-hard in general.

To overcome this challenge, the problem is often reformulated using a factorized representation, commonly referred to as the Burer-Monteiro (BM) decomposition [19]. Specifically, the low-rank matrix M is parameterized as $M = XX^\top$, where $X \in \mathbb{R}^{n\times r}$. Substituting this parameterization into the objective function transforms the optimization problem into:

$$\min_{X\in\mathbb{R}^{n\times r}} h(X) = f(XX^\top) = \frac{1}{2}\|\mathcal{A}(XX^\top) - b\|_2^2. \tag{3}$$

This reformulation eliminates the explicit rank constraint but introduces additional non-convexity arising from the bilinear structure of $XX^\top$.

2.1 Restricted Isometry Property (RIP)

The Restricted Isometry Property (RIP) [11] is a widely used condition in low-rank matrix recovery that ensures the linear measurement operator $\mathcal{A}$ preserves the geometry of low-rank matrices. Formally, $\mathcal{A}$ satisfies the RIP of rank $r \le p$ with constant $\delta_p \in (0,1)$ if, for any $M \in \mathbb{R}^{n\times n}$, the following inequality holds:

$$(1-\delta_p)\|M\|_F^2 \le \|\mathcal{A}(M)\|_2^2 \le (1+\delta_p)\|M\|_F^2, \tag{4}$$

where $\|M\|_F$ denotes the Frobenius norm of M. Intuitively, the RIP ensures that the measurement operator $\mathcal{A}$ does not significantly distort the distances between low-rank matrices, which is essential for accurate recovery.

To address the limitations of the RIP framework, we adopt an alternative formulation based on the (L_s, p)-Restricted Strong Smoothness (RSS) and (α_s, p)-Restricted Strong Convexity (RSC) properties [20]. The formal definition of these properties is provided in Definition A.10.

These conditions separately characterize the smoothness and curvature of the measurement operator $\mathcal{A}$, where $L_s = 1+\delta_p$ and $\alpha_s = 1-\delta_p$. This formulation offers a more flexible and expressive generalization of the RIP. Notably, the parameters L_s and α_s play a crucial role in the subsequent derivations.

2.2 Tensor Lifting for Matrix Sensing

To facilitate the understanding of tensor lifting methods for matrix sensing, we first recommend readers to review the foundational concepts of tensor algebra provided in Appendix A.1, as these concepts are crucial for the subsequent discussions in this work.

Tensor lifting reformulates the matrix sensing problem by embedding the matrix variable $X \in \mathbb{R}^{n\times r}$ into a higher-dimensional tensor space. Specifically, [10, 21] define the lifted tensor as $\mathbf{w} = \mathrm{vec}(X)^{\otimes l} \in \mathbb{R}^{nr \circ l}$, where l denotes the lifting order. To preserve the structure of the original matrix during lifting and recovery, they introduce a permutation tensor $\mathbf{P} \in \mathbb{R}^{n\times r\times nr}$ that satisfies:

$$\langle \mathbf{P}, \mathrm{vec}(X)\rangle_3 = X, \quad \forall X \in \mathbb{R}^{n\times r}, \tag{5}$$

where $\langle\cdot,\cdot\rangle_3$ represents contraction along the third mode of $\mathbf{P}$. This mapping guarantees that any vectorized matrix $\mathrm{vec}(X)$ can be faithfully reconstructed from the tensor space using $\mathbf{P}$. The sensing operator $\mathcal{A}$ is then extended to the tensor space. The original operator $\mathbf{A} \in \mathbb{R}^{m\times n\times n}$, defined as $\mathbf{A}_{ijk} = (A_i)_{jk}$ for $i \in [m]$ and $(j,k) \in [n]\times[n]$, is lifted to $\mathbf{A}^{\otimes l}$, enabling it to act on higher-dimensional tensors. Using the lifted operator and variable, the reconstructed matrix is expressed as $\mathbf{P}(\mathbf{w}) = \langle \mathbf{P}^{\otimes l}, \mathbf{w}\rangle_{3*[l]}$, where $\langle\cdot,\cdot\rangle_{3*[l]}$ generalizes mode-wise contractions for order-l tensors.

The optimization problem in tensor space is formulated as:

$$\min_{\mathbf{w}\in\mathbb{R}^{nr\circ l}} \left\|\left\langle \mathbf{A}^{\otimes l}, \langle \mathbf{P}(\mathbf{w}), \mathbf{P}(\mathbf{w})\rangle_{2*[l]}\right\rangle - b^{\otimes l}\right\|_F^2, \tag{6}$$

where $\langle \mathbf{P}(\mathbf{w}), \mathbf{P}(\mathbf{w})\rangle_{2*[l]}$ contracts modes $2, 4, \ldots, 2l$ to retain the bilinear structure of the original matrix variable.

To simplify notation and clarify the problem structure, two auxiliary functions are defined. The reconstruction error in the tensor space is expressed as:

$$f^l(\mathbf{M}) := \left\|\left\langle \mathbf{A}^{\otimes l}, \mathbf{M}\right\rangle - b^{\otimes l}\right\|_F^2, \tag{7}$$

and the lifted objective function is given by:

$$h^l(\mathbf{w}) := f^l\left(\langle \mathbf{w}, \mathbf{w}\rangle_{2*[l]}\right), \tag{8}$$

where $\mathbf{M}$ is a candidate tensor, and $\langle \mathbf{w}, \mathbf{w}\rangle_{2*[l]}$ denotes the contraction of $\mathbf{w}$ with itself. These definitions establish a clear connection between the lifted tensor space and the original matrix space.

Tensor lifting fundamentally transforms the optimization landscape by *converting spurious local minima in the matrix space into strict saddle points in the tensor space*. Gradient-based methods can leverage this favorable geometry to escape saddle points and converge to global minima. However, the computational cost of tensor lifting increases geometrically with the lifting order l, making it challenging for large-scale problems. Developing efficient implementations and approximations is therefore critical for practical applications.

2.3 Implicit Regularization in Over-parameterized Optimization

An intriguing phenomenon in over-parameterized optimization is *implicit regularization*, where optimization algorithms, such as gradient descent, exhibit a natural bias toward solutions with desirable properties, even in the absence of explicit constraints. For instance, in the Burer-Monteiro (BM) decomposition for matrix sensing, gradient descent inherently tends to find low-rank solutions [22], effectively acting as an implicit form of regularization. Similarly, in tensor lifting, the optimization dynamics in the tensor space bias the iterates toward tensors that are approximately rank-1 [21], which correspond to low-rank solutions in the matrix space. This property has been exploited to analyze the convergence behavior of gradient descent in the tensor space. In our work, this approximate rank-1 structure enables us to establish a connection between the gradient descent trajectories in the tensor space and those in the matrix space. Consequently, escape directions identified in the tensor space can serve as an *oracle* for guiding escape directions in the matrix space.

2.4 Limitations of Full Tensor Lifting and Solutions

Despite its theoretical benefits, full tensor lifting faces significant practical challenges. The computational cost grows exponentially with the lifting order l, as the size of the lifted tensor $\mathbf{w}$ scales exponentially. Additionally, the increased number of variables and constraints complicates optimization, often requiring more iterations to converge.

To address these issues, we propose the LEAC (Lift-Escape-And-Collapse) framework, which includes two variants: LEAC-Canonical and LEAC-Simulated. LEAC-Canonical explicitly performs tensor lifting to escape spurious local minima, while LEAC-Simulated avoids explicit tensor operations by simulating the escape dynamics directly in the matrix space. LEAC-Canonical serves as an intermediary algorithm to demonstrate the philosophy of our approach, while LEAC-Simulated is our main result. The simulation retains the benefits of over-parameterization while significantly reducing computational overhead. Consequently, LEAC-Simulated is particularly effective for large-scale problems where full tensor lifting is infeasible.

2.5 Algorithms that (Might) Escape Local Minima

Particle Swarm Optimization (PSO), inspired by natural predation processes, conducts stochastic searches to find optimal solutions [23]. Stochastic Gradient Descent (SGD), on the other hand, introduces randomness by using a subset of data to compute gradients at each step [24]. Momentum-based and adaptive SGD further augments the ability to escape by incorporating a weighted average of past and current gradients [25, 26, 5]. However, none of these methods can escape local minima deterministically [27], and escapes cannot be expected nor actively initiated.

2.6 Notations

Let $\hat{X}$ represent a local minimum in the matrix space (3), and let $\hat{\mathbf{w}}$ denote its corresponding strict saddle point in the tensor space. Throughout this work, we frequently refer to decompositions of $\hat{X}$ and $\nabla f(\hat{X}\hat{X}^\top)$:

$$\hat{X} = \sum_{\phi=1}^{r} \sigma_\phi v_\phi q_\phi^\top, \;\; \nabla f(\hat{X}\hat{X}^\top) = \sum_{\theta=1}^{n} \lambda_\theta u_\theta u_\theta^\top \tag{9}$$

where σ_r, v_r, and q_r represent the smallest singular value and its corresponding left and right singular vectors, respectively. Similarly, λ_n and u_n denote the smallest eigenvalue and its eigenvector.

Additionally, we use the following notations: $\circ l$ denotes the shorthand for the l-fold Cartesian product; $[l]$ denotes the integer set $1, 2, \ldots, l$ for a positive integer l.

3 Deterministic Escape with Canonical Over-Parametrization

To better facilitate the understanding of our new approach, we first present the LEAC-Canonical algorithm in this section, which has a more intuitive structure. We now outline the theoretical foundations of this approach, followed by a description of the algorithm derived from these results.

3.1 Theoretical Foundations of LEAC-Canonical

The LEAC-Canonical mechanism relies on two key theoretical results that establish the connection between spurious local minima in the matrix space and strict saddle points in the tensor space. These results also provide a structured escape direction that enables overcoming such saddle points.

The first theorem demonstrates that under certain conditions, a spurious local minimum $\hat{X}$ in the matrix space becomes a strict saddle point in the tensor space after tensor lifting. Moreover, it identifies a rank-1 escape direction in the tensor space, constructed using the smallest singular vector q_r of $\hat{X}$ and the smallest eigenvector u_n of $\nabla f(\hat{X}\hat{X}^\top)$.

Theorem 3.1 (informal, adapted from [10]). *Consider a spurious local minimum $\hat{X} \in \mathbb{R}^{n\times r}$ of Equation* (3) *with rank $r < n$, where $r_{search} = r$ and $\hat{X}\hat{X}^\top \neq M^*$. Assume that $\mathcal{A}$ satisfies the RSC and RSS conditions. Then, the lifted tensor $\hat{\mathbf{w}} = \operatorname{vec}(\hat{X})^{\otimes l}$ is a strict saddle point of Equation* (6) *with a rank-1 symmetric escape direction* $\operatorname{vec}(u_n q_r^\top)^{\otimes l}$*, provided that $\hat{X}$ satisfies:*

$$-\lambda_n \geq \frac{\alpha_s \|\hat{X}\hat{X}^\top - M^*\|_F^2}{2\operatorname{Tr}(M^*)} \geq \frac{L_s \sigma_r^2}{2}, \tag{10}$$

if l is odd and sufficiently large. $-\lambda_n$ denotes the smallest eigenvalue of $\nabla f(\hat{X}\hat{X}^\top)$, and σ_r is the r-th singular value of $\hat{X}$.

The second theorem guarantees that moving along the escape direction in the tensor space always reduces the objective value, regardless of the step size η.

Theorem 3.2 (Tensor Objective Descent Along the Escape Direction). *For any step size $\eta > 0$, moving along the escape direction* $\operatorname{vec}(u_n q_r^\top)^{\otimes l}$ *reduces the tensor space objective value in Equation* (8)*:*

$$h^l(\mathbf{w}_{escape}) = h^l(\hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}) < h^l(\hat{\mathbf{w}}). \tag{11}$$

These two theorems form the theoretical foundation of the LEAC-Canonical mechanism. By lifting the optimization problem to the tensor space and updating along a structured escape direction, the mechanism ensures descent in the tensor space objective, enabling deterministic escape from spurious local minima.

3.2 Algorithm Description of LEAC-Canonical

Based on the theoretical results outlined above, the LEAC-Canonical mechanism is implemented in Algorithm 1. The algorithm begins by identifying a spurious local minimum $\hat{X}$ in the matrix space and lifting it to the tensor space through the tensorized vectorization operator. In the tensor space, the algorithm updates the lifted variable along the escape direction. Finally, the updated tensor is collapsed back into the matrix space using TensorPCA [21], producing the escape point $\check{X}$.

Algorithm 1 LEAC with Canonical Over-Parametrization

```
1: Input: Local minimum $\hat{X}$, lifting order $l$, step size $\eta$
2: Output: Escape point $\check{X} \in \mathbb{R}^{n\times r}$
3: Perform SVD on $\hat{X}$: $\hat{X} = \sum_{\phi=1}^{r} \sigma_\phi v_\phi q_\phi^\top$
4: Compute the eigendecomposition of $\nabla f(\hat{X}\hat{X}^\top)$: $\nabla f(\hat{X}\hat{X}^\top) = \sum_{\theta=1}^{n} \lambda_\theta u_\theta u_\theta^\top$
5: Lift $\hat{X}$ to the tensor space: $\hat{\mathbf{w}} = \operatorname{vec}(\hat{X})^{\otimes l}$
6: Update the lifted variable in the tensor space: $\check{\mathbf{w}} = \hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}$
7: Collapse the lifted tensor back to matrix space: $\check{X} = \mathbf{TensorPCA}(\check{\mathbf{w}})$
```

While the LEAC-Canonical mechanism is theoretically effective for escaping spurious local minima, it has notable limitations. First, although the escape direction guarantees descent in the tensor space objective, ensuring a corresponding descent in the matrix space objective is challenging, as the step size η must be carefully calibrated. Second, the tensor update $\hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}$ often produces a tensor that is not rank-1, necessitating the use of TensorPCA to extract the dominant rank-1 component. Finally, the lifting and collapsing processes introduce significant computational overhead, which restricts the scalability of the method in high-dimensional settings. To overcome these challenges, the next section introduces the LEAC-Simulated mechanism, which leverages simulated over-parameterization to bypass explicit lifting and collapsing while retaining the benefits of over-parameterization.

4 Deterministic Escape with Simulated Over-Parametrization

4.1 Theoretical Foundations of LEAC-Simulated

4.1.1 Correspondence Between Tensor and Matrix Trajectories

This subsection establishes the theoretical connection between optimization trajectories in the matrix space and their counterparts in the tensor space. Specifically, we demonstrate a one-to-one correspondence between gradient descent (GD) trajectories in the two spaces and provide guarantees that tensor-space solutions near the escape point correspond to approximate rank-1 tensors aligned with matrix-space solutions. These results are crucial for motivating the design of closed-form approximations for rank-1 tensor escape points (detailed in the next subsection).

Theorem 4.1 (One-to-One Correspondence Between Matrix and Tensor Trajectories)**.** *Let $X_0 = \epsilon X \in \mathbb{R}^{n\times r_{search}}$ represent the initialization in the matrix space, where $\|X\|_F^2 = 1$ and $r_{search} > r$. Denote the matrix-space gradient descent trajectory by $\{X_t\}_{t=0}^{\infty}$, where the updates follows $X_{t+1} = X_t - \eta^{\frac{1}{l}}\nabla h(X_t)$. Let $\mathbf{w}_0 = \operatorname{vec}(X_0)^{\otimes l}$ be the corresponding initialization in the tensor space. The tensor-space gradient descent trajectory, denoted by $\{\mathbf{w}_t\}_{t=0}^{\infty}$, follows the updates $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\nabla h^l(\mathbf{w}_t)$, with an arbitrary step size η. Then, for sufficiently small ϵ, the following holds for all $t \geq 0$:*

$$\operatorname{vec}(X_t)^{\otimes l} \approx \mathbf{w}_t, \tag{12}$$

where $\mathbf{w}_t$ is approximately rank-1. Moreover, this correspondence between $\{\mathbf{w}_t\}_{t=0}^{\infty}$ and $\{X_t\}_{t=0}^{\infty}$ is one-to-one.

Approximation Note: *The term "approximately rank-1" is defined as being κ-rank-1 with a reasonably small κ, where κ quantifies the ratio between the second-largest and largest "v-eigenvalues" of $\mathbf{w}_t$. For a more detailed definition, please refer to Theorem A.9.*

This theorem establishes a direct link between gradient descent trajectories in the matrix space and their lifted counterparts in the tensor space. It shows that as long as the initialization ϵ is sufficiently small, the tensor-space trajectory $\{\mathbf{w}_t\}_{t=0}^{\infty}$ remains close to the lifted versions of the matrix-space trajectory $\{\operatorname{vec}(X_t)^{\otimes l}\}_{t=0}^{\infty}$, thereby preserving the approximate rank-1 structure of the tensors.

Corollary 4.2 (Escape Direction Correspondence Near Approximate Rank-1 Solutions)**.** *Under the same conditions as Theorem 4.1, consider a strict saddle point $\hat{\mathbf{w}}$ in the tensor space and its escape point along the direction $\operatorname{vec}(u_n q_r^\top)^{\otimes l}$:*

$$\mathbf{w}_{escape} = \hat{\mathbf{w}} + \eta\operatorname{vec}(u_n q_r^\top)^{\otimes l}, \tag{13}$$

where $\eta > 0$. Then, there exists an approximately rank-1 tensor $\check{\mathbf{w}}$ near $\mathbf{w}_{escape}$, such that $\check{\mathbf{w}}$ corresponds to a point along a matrix-space gradient descent trajectory $\{X_t\}_{t=0}^{\infty}$.

Interpretation: *The escape point $\mathbf{w}_{escape}$ lies near an approximate rank-1 tensor that can be mapped back to the matrix space. This is because $\mathbf{w}_{escape}$ either remains on the original matrix-tensor GD trajectory or transitions to a new one. By Theorem 4.1, all points along GD trajectories in the tensor space have an approximate rank-1 correspondence in the matrix space.*

This corollary extends Theorem 4.1 by ensuring that any tensor-space escape point along the strict saddle's escape direction remains close to an approximate rank-1 tensor. This guarantees that the escape point can be effectively associated with a matrix-space solution, motivating the design of the closed-form structure for $\check{\mathbf{w}}$.

4.1.2 Closed-Form Structure of Rank-1 Tensor

In this subsection, we present the closed-form structure of $\check{\mathbf{w}}$, a strict rank-1 tensor that approximates the tensor-space escape point $\mathbf{w}_{\text{escape}}$ in terms of the tensor spectral norm. This result offers a computationally efficient representation of escape points without requiring a full transition to the tensor space, significantly reducing computational complexity while preserving theoretical guarantees.

Theorem 4.3 (Closed-form Structure of $\check{\mathbf{w}}$)**.** *The strict rank-1 tensor $\check{\mathbf{w}} = \operatorname{vec}(\check{X})^{\otimes l}$, which approximates the tensor-space escape point $\mathbf{w}_{escape}$, is given by:*

$$\check{X} = \sum_{\phi=1}^{r-1}\sigma_\phi v_\phi q_\phi^\top + \sigma_r(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)q_r^\top. \tag{14}$$

If $\sigma_r^l\alpha^{\frac{l}{2}} \to \eta$ and $\beta^{\frac{l}{2}} \to 1$, then

$$\left\|\operatorname{vec}(\check{X})^{\otimes l} - \left[\hat{\mathbf{w}} + \eta\operatorname{vec}(u_n q_r^\top)^{\otimes l}\right]\right\|_S \to 0, \tag{15}$$

where $\|\cdot\|_S$ denotes the tensor spectral norm. In other words, the strict rank-1 tensor $\mathrm{vec}(\check{X})^{\otimes l}$ *closely approximates the tensor* $\mathbf{w}_{escape}$ *in the spectral norm sense.*

The construction of $\check{X}$ is computationally efficient, as it requires only the singular value decomposition (SVD) of X, the eigendecomposition of $\nabla f(XX^\top)$, and the parameters α and β. The significance of this result lies in its ability to bypass the costly procedures of the LEAC-Canonical mechanism, which involves fully lifting to the tensor space, performing deterministic escape, and subsequently collapsing back to the matrix space via computationally demanding procedures such as TensorPCA. In contrast, the closed-form structure of $\check{\mathbf{w}}$ allows us to directly identify the matrix-space point corresponding to the approximate rank-1 tensor in the tensor space.

Based on Theorem 4.3, by keeping $\beta = 1$ and tuning the parameter α, we can control the tensor-space escape step size η to ensure deterministic escape. In the next subsection, we will define the range of valid α values that guarantee $h(\check{X}) < h(\hat{X})$, providing theoretical assurance for matrix-space escape and practical guidance for parameter selection.

4.1.3 Conditions for Escaping Local Minima in the Matrix Space

In this subsection, we establish sufficient conditions for escaping local minima entirely within the matrix space. This result complements the closed-form solution presented in the previous subsection, enabling the determination of an appropriate escape step size in the matrix space, controlled by the parameter α, without requiring full lifting to the tensor space.

Theorem 4.4 (Sufficient Condition for Matrix-Space Escape)**.** *Assume that $\mathcal{A}$ satisfies the RSS condition. For a matrix $\hat{X}$ that is a first-order critical point (FOP) of $h(X)$, the sufficient condition for $h(\check{X}) < h(\hat{X})$ is:*

$$\alpha \in \left(0, \frac{-2\lambda_n}{L_s \sigma_r^2} - 2\right). \tag{16}$$

Here, L_s is the RSS constant. Importantly, this condition is independent of the tensor lift level l and applies directly in the matrix space.

Intersection Note: *To ensure that Equation* (16) *admits a valid solution for α, Equation* (10) *must hold strictly. This is because, otherwise, the critical point in the tensor space would not be a strict saddle point, and the escape direction* $\mathrm{vec}(u_n q_r^\top)^{\otimes l}$ *in the tensor space would provide no meaningful guidance for escaping from the matrix space.*

Theorem 4.4 provides a sufficient condition for escaping local minima in the matrix space. When paired with the closed-form structure of $\check{\mathbf{w}}$ established in Theorem 4.3, this result enables direct computation of the escape step size in the matrix space by identifying an appropriate range for α.

By effectively bridging the matrix and tensor spaces while avoiding the need for explicit lifting to the tensor space, Theorem 4.4 consolidates the computational efficiency of the LEAC mechanism. This approach allows the exploitation of the benefits of tensor-space lifting without incurring the full computational cost associated with such operations.

Algorithm 2 LEAC with Simulated Over-Parametrization

```
1: Input: Local minimum $\hat{X}$, lifting order $l$, step size $\eta$, Restricted Strong Smoothness constant $L_s$
2: Output: Escape point $\check{X} \in \mathbb{R}^{n\times r}$
3: Perform SVD on $\hat{X}$: $\hat{X} = \sum_{\phi=1}^{r} \sigma_\phi v_\phi q_\phi^\top$
4: Compute the eigendecomposition of $\nabla f(\hat{X}\hat{X}^\top)$: $\nabla f(\hat{X}\hat{X}^\top) = \sum_{\theta=1}^{n} \lambda_\theta u_\theta u_\theta^\top$
5: Compute upper bound for $\alpha$: $\Upsilon = \frac{-2\lambda_n}{L_s\sigma_r^2} - 2$
6: while True do
7:     Assign $\beta = 1$ and Sample $\alpha$ from $(0, \Upsilon)$
8:     Compute rank-1 approximation $\check{X}$ using Equation 14
9:     if $h(\check{X}) < h(\hat{X})$ then
10:        break
11:    end if
12: end while
```

4.2 Algorithm Description of LEAC-Simulated

The LEAC-Simulated mechanism leverages simulated over-parameterization to efficiently escape spurious local minima in the matrix space.

The algorithm starts at a spurious local minimum $\hat{X}$ in the matrix space and identifies an escape direction (analogous to "oracle") that derived from the tensor space. Then, based on the closed-form structure of approximate rank-1 tensors established in Theorem 4.3, LEAC-Simulated constructs an escape point $\check{X}$ entirely within the matrix space. By iteratively refining the parameters α and β based on Theorem 4.4, the algorithm ensures that the matrix-space escape point $\check{X}$ satisfies the descent condition $h(\check{X}) < h(\hat{X})$. The complete procedure is summarized in Algorithm 2.

Unlike the LEAC-Canonical mechanism, which requires explicit lifting to the tensor space and subsequent collapsing back to the matrix space, LEAC-Simulated avoids these computationally intensive procedures by directly exploiting the theoretical correspondence between tensor-space and matrix-space trajectories. This approach preserves the benefits of over-parameterization while significantly reducing computational complexity, making it scalable to high-dimensional settings.

5 Numerical Experiments

In this section, we evaluate the proposed LEAC-Simulated mechanism on the Perturbed Matrix Completion [9] problem, which is known to contain an exponential number of spurious solutions. The experiments aim to validate the effectiveness of LEAC-Simulated in escaping spurious local minima and its efficiency by comparing the number of optimization iterations required until convergence.

5.1 Perturbed Matrix Completion Setup

The perturbed matrix completion problem involves recovering a ground-truth low-rank matrix $M^* \in \mathbb{R}^{n\times n}$ with $\operatorname{rank}(M^*) = r$, given noisy observations generated by a perturbed linear operator $\mathcal{A}_\rho$. The operator $\mathcal{A}_\rho$ is defined as:

$$\mathcal{A}_\rho(M_{ij}) = \begin{cases} M_{ij}, & \text{if } (i,j) \in \Omega, \\ \rho M_{ij}, & \text{otherwise}, \end{cases} \tag{17}$$

where Ω is the measurement set, specified as:

$$\Omega = \{(i,i), (i,2k), (2k,i) \,|\, \forall i \in [n], k \in [\lfloor n/2 \rfloor]\}, \tag{18}$$

and ρ is a small perturbation factor (set to $\rho = 0.01$). The task is to recover M^* from these noisy observations while avoiding spurious local minima, which are known to exist in this problem class due to the incomplete and perturbed nature of the measurements.

Following the experimental protocol in [21], we set the lifted level $l = 3$ and initialize the optimization with $X_0 = \epsilon X$, where X is sampled from a standard Gaussian distribution and $\epsilon = 10^{-7}$. Each experiment is repeated over 64 independent trials to ensure statistical robustness, and performance is averaged across these trials.

5.2 Baseline Algorithms

To assess the performance of LEAC-Simulated, we compare it against the following baseline solvers:

- *Unlift Solver*: This method performs standard gradient descent entirely within the matrix space without any form of over-parameterization. It serves as a baseline to assess the advantages of leveraging over-parameterization.
- *Lift Solver*: This method conducts gradient descent fully in the tensor space, explicitly lifting and optimizing in the over-parameterized space. It represents the canonical tensor-based optimization approach and serves as a computationally intensive baseline.

Our solver, referred to as the *Semilift Solver*, adopts a hybrid approach that combines matrix and tensor space optimization. Specifically, LEAC-Simulated primarily operates in the matrix space using simulated over-parameterization. However, in cases where the strict saddle condition (Equation (10)) is not satisfied, the algorithm explicitly lifts to the tensor space and applies gradient descent directly in the tensor space. This ensures that LEAC-Simulated maintains its theoretical guarantees even in challenging scenarios.

5.3 Key Observations and Results

Figure 1 illustrates the gradient norm and loss curves for the Unlift Solver, Lift Solver, and Semilift Solver. As shown, both the Unlift Solver and Semilift Solver experience little to no loss reduction during the first 50–100 epochs. This

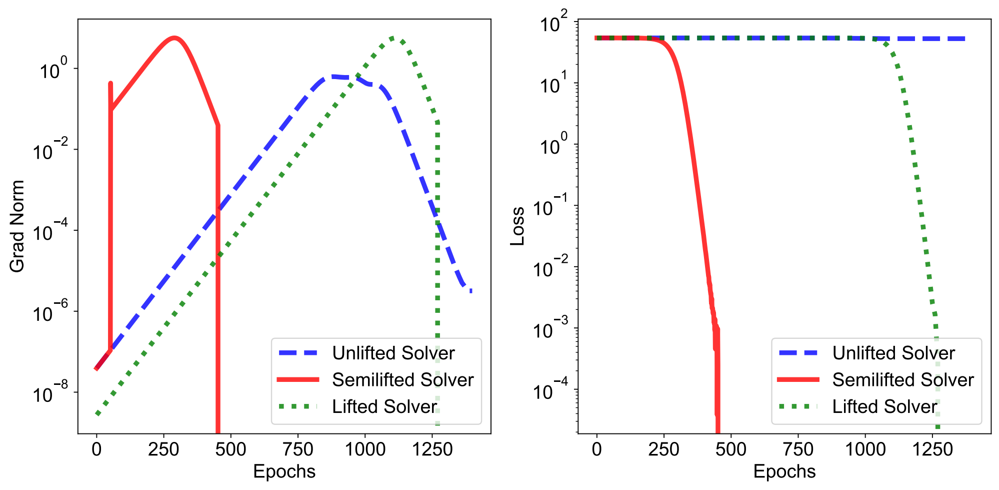

Figure 1: Gradient norm (left) and loss curves (right) for Unlift Solver, Lift Solver, and Semilift Solver (**ours**) over 1400 epochs.

suggests that both solvers encounter a spurious local minimum $\hat{X}$ in the matrix space, while the Lift Solver hits the corresponding strict saddle $\hat{\mathbf{w}}$ in the tensor space.

After this initial stagnation, the Semilift Solver successfully escapes the spurious local minimum $\hat{X}$ by leveraging the LEAC-Simulated mechanism, jumping directly to $\check{X}$. This escape is immediately reflected in the loss curve, where the loss starts to decrease rapidly, and in the gradient norm curve, which exhibits a sharp rise followed by a steep drop. In contrast, the Unlift Solver, lacking the guidance provided by over-parameterization, fails to escape the local minimum despite a temporary rise and fall in the gradient norm. The loss curve for the Unlift Solver remains relatively flat, indicating that the algorithm is trapped near the spurious local minimum.

The Lift Solver, operating entirely in the tensor space, also successfully escapes the strict saddle and eventually achieves a loss reduction comparable to the Semilift Solver. However, this comes at the cost of significantly more iterations. As indicated by Theorem 4.1, tensor-space gradient descent updates with step size η are approximately equivalent to matrix-space updates with step size $\eta^{\frac{1}{l}}$. This smaller effective step size in the matrix space results in a slower convergence rate for the Lift Solver, requiring many more iterations to escape the local minimum.

By contrast, the Semilift Solver avoids this inefficiency by directly jumping from $\hat{X}$ to $\check{X}$ in a single step, leveraging the closed-form structure of approximate rank-1 tensors provided by the LEAC-Simulated mechanism. This significantly reduces the number of iterations required for escape while maintaining theoretical guarantees.

Remark: The loss curves also reveal that the initialization point is already near a spurious local minimum, making the Lift Solver's behavior analogous to applying the LEAC-Canonical mechanism (lifting to the tensor space near the starting point to escape). Comparing the Lift Solver and Semilift Solver thus provides an indirect comparison between the LEAC-Canonical and LEAC-Simulated mechanisms. The results highlight the computational efficiency of LEAC-Simulated, as it achieves similar escape performance with substantially fewer iterations.

Solver ($n = 11$)	Unlift	Semilift (**ours**)	Lift
Success Rate ↑	0.05	**0.84**	0.82
Convergence Epoch ↓	1364	**512**	1313

Table 1: Comparison of Success Rate and Convergence Epoch for $n = 11$ (n is the size of M^*).

We further quantify the performance of the three solvers by comparing their success rates and convergence speeds, as summarized in Table 1. Here, the *Success Rate* is defined as the proportion of 64 trials where the final solution satisfies $\|X_T X_T^\top - M^*\|_F < 0.05$, and the *Convergence Epoch* (T) is the number of epochs required to reach this solution. The reason we use success rate instead of average reconstruction error is that failed trails will have very large errors, skewing the statistics by a large margin, distracting people from the true message.

The results in Table 1 further demonstrate the advantages of the LEAC-Simulated mechanism. Specifically, the Semilift Solver achieves a success rate of 0.84, outperforming the Unlift Solver significantly (0.05) and slightly surpassing the

Lift Solver (0.82). This highlights the robustness of LEAC-Simulated in escaping spurious local minima and converging to the correct solution.

Moreover, the Semilift Solver requires only 512 epochs on average to reach the solution, which is a substantial improvement over the Lift Solver (1313 epochs) and the Unlift Solver (1364 epochs). This result aligns with our earlier observations in Figure 1, where the Semilift Solver leverages the efficiency of simulated over-parameterization to escape local minima in a single step and converges significantly faster than its counterparts.

6 Conclusion

In this work, we introduced LEAC (Lift, Escape, Approximate, and Collapse), a deterministic framework for escaping spurious local minima in non-convex matrix sensing. By leveraging simulated over-parameterization, LEAC has the ability to actively calculate a direction of guaranteed descent when trapped in local minima. This work can serve as an initial step to construct general first-order optimizers with the ability of deterministic escape for more complex machine learning models.

Acknowledgments

This work was supported by the City University of Hong Kong (# 9382001).

References

[1] Katta G Murty and Santosh N Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[2] Yuejie Chi, Yue M Lu, and Yuxin Chen. Nonconvex optimization meets low-rank matrix factorization: An overview. *IEEE Transactions on Signal Processing*, 67(20):5239–5269, 2019.

[3] Ju Sun, Qing Qu, and John Wright. Escaping from saddle points on riemannian manifolds. *arXiv preprint arXiv:1901.06037*, 2019.

[4] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International conference on machine learning*, pages 1724–1732. PMLR, 2017.

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[6] Matthew Staib, Sashank Reddi, Satyen Kale, Sanjiv Kumar, and Suvrit Sra. Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, pages 5956–5965. PMLR, 2019.

[7] Yuanzhe Tao, Huizhuo Yuan, Xun Zhou, Yuan Cao, and Quanquan Gu. Towards simple and provable parameter-free adaptive gradient methods. *arXiv preprint arXiv:2412.19444*, 2024.

[8] Richard Y Zhang. Improved global guarantees for the nonconvex burer–monteiro factorization via rank overparameterization. *arXiv preprint arXiv:2207.01789*, 2022.

[9] Baturalp Yalcin, Ziye Ma, Javad Lavaei, and Somayeh Sojoudi. Semidefinite programming versus burer-monteiro factorization for matrix sensing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[10] Ziye Ma, Igor Molybog, Javad Lavaei, and Somayeh Sojoudi. Over-parametrization via lifting for low-rank matrix sensing: Conversion of spurious solutions to strict saddle points. In *International Conference on Machine Learning*. PMLR, 2023.

[11] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[12] Emmanuel J Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[13] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[15] Yi-Kai Liu. Universal low-rank matrix recovery from pauli measurements. *Advances in Neural Information Processing Systems*, 24, 2011.

[16] Hanxi Li, Chunhua Shen, and Qinfeng Shi. Real-time visual tracking using compressive sensing. In *CVPR 2011*, pages 1305–1312. IEEE, 2011.

[17] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pages 2–47. PMLR, 2018.

[18] Igor Molybog, Ramtin Madani, and Javad Lavaei. Conic optimization for quadratic regression under sparse noise. *The Journal of Machine Learning Research*, 21(1):7994–8029, 2020.

[19] Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[20] Prateek Jain, Purushottam Kar, et al. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–363, 2017.

[21] Ziye Ma, Javad Lavaei, and Somayeh Sojoudi. Algorithmic regularization in tensor optimization: Towards a lifted approach in matrix sensing. *Advances in Neural Information Processing Systems*, 36, 2024.

[22] Dominik Stöger and Mahdi Soltanolkotabi. Small random initialization is akin to spectral learning: Optimization and generalization guarantees for overparameterized low-rank matrix reconstruction. *Advances in Neural Information Processing Systems*, 34:23831–23843, 2021.

[23] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[24] Herbert E. Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[25] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12(1):145–151, 1999.

[26] Tijmen Tieleman and Geoffery Hinton. Rmsprop gradient optimization. *URL http://www. cs. toronto. edu/tijmen/csc321/slides/lecture_slides_lec6. pdf*, 2014.

[27] Sebastian Stich and Harsh Harshvardhan. Escaping local minima with stochastic noise. In *Neurips Workshop on Optimization for Machine Learning*, 2021.

[28] Pierre Comon, Gene Golub, Lek-Heng Lim, and Bernard Mourrain. Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1254–1279, 2008.

[29] Tamara G Kolda. Numerical optimization for symmetric tensor decomposition. *Mathematical Programming*, 151(1):225–248, 2015.

[30] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[31] Liqun Qi, Shenglong Hu, Xinzhen Zhang, and Yannan Chen. Tensor norm, cubic power and gelfand limit. *arXiv preprint arXiv:1909.10942*, 2019.

[32] Lek-Heng Lim and Pierre Comon. Blind multilinear identification. *IEEE Transactions on Information Theory*, 60(2):1260–1280, 2013.

[33] Liqun Qi. The spectral theory of tensors (rough version). *arXiv preprint arXiv:1201.3424*, 2012.

[34] Wooseok Ha, Haoyang Liu, and Rina Foygel Barber. An equivalence between critical points for rank constraints versus low-rank factorizations. *SIAM Journal on Optimization*, 30(4):2927–2955, 2020.

[35] Haixiang Zhang, Yingjie Bi, and Javad Lavaei. General low-rank matrix optimization: Geometric analysis and sharper bounds. *Advances in Neural Information Processing Systems*, 34:27369–27380, 2021.

[36] Ziye Ma, Ying Chen, Javad Lavaei, and Somayeh Sojoudi. Absence of spurious solutions far from ground truth: A low-rank analysis with high-order losses. In *International Conference on Artificial Intelligence and Statistics*, volume 238. PMLR, 2024.

[37] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[38] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

A Preliminary Knowledge

A.1 Tensor Algebra Basics

Definition A.1 (Tensor)**.** As a generalization of how vectors parametrize finite-dimensional vector spaces, tensors are parametrized using arrays generated from the product of finite-dimensional vector spaces, as described in [28]. Specifically, an l-way array is defined as:

$$\mathbf{s} = \{s_{i_1 i_2 \dots i_l} \mid 1 \le i_k \le n_k, 1 \le k \le l\} \in \mathbb{R}^{n_1 \times \cdots \times n_l}. \tag{19}$$

In this paper, tensors and arrays are considered synonymous, as there exists an isomorphism between them. Furthermore, if $n_1 = \cdots = n_l$, we refer to the tensor (or array) as an l-order (way), n-dimensional tensor. For ease of representation, we use the notation $\mathbb{R}^{n \circ l}$, where $n \circ l := n \times \cdots \times n$ (repeated l times). Throughout this work, tensors are denoted by bold variables, while matrices, vectors, and scalars are represented in other fonts unless stated otherwise.

Definition A.2 (Symmetric Tensor)**.** Analogous to the definition of symmetric matrices, an order-l tensor $\mathbf{s}$ with equal dimensions (i.e., $n_1 = \cdots = n_l$, also called a cubic tensor) is said to be symmetric if its entries remain invariant under any permutation of their indices:

$$s_{i_{\sigma(1)} \cdots i_{\sigma(l)}} = s_{i_1 \cdots i_l} \quad \forall \sigma,\ i_1, \dots, i_l \in \{1, \dots, n\}, \tag{20}$$

where $\sigma \in \mathcal{G}_l$ represents a specific permutation, and $\mathcal{G}_l$ is the symmetric group of permutations on $\{1, \dots, l\}$. The set of symmetric tensors is denoted as $\mathcal{S}^l(\mathbb{R}^n)$.

Definition A.3 (Rank of Tensors)**.** The rank of a cubic tensor $\mathbf{s} \in \mathbb{R}^{n \circ l}$ is defined as:

$$\operatorname{rank}(\mathbf{s}) = \min\{r \mid \mathbf{s} = \sum_{i=1}^{r} a_i \otimes b_i \otimes \cdots \otimes c_i\}, \tag{21}$$

for some vectors $a_i, b_i, \dots, c_i \in \mathbb{R}^n$. Furthermore, as noted in [29], if $\mathbf{s}$ is a symmetric tensor, it can be decomposed as:

$$\mathbf{s} = \sum_{i=1}^{r} \lambda_i a_i \otimes \cdots \otimes a_i := \sum_{i=1}^{r} \lambda_i a_i^{\otimes l}, \tag{22}$$

where the rank is defined as the number of nonzero λ_i's, analogous to the rank of symmetric matrices. A key concept in this work is that of rank-1 tensors. For any tensor $\mathbf{s}$, a necessary and sufficient condition for it to be rank-1 is:

$$\mathbf{s} = a^{\otimes l}, \tag{23}$$

for some $a \in \mathbb{R}^n$.

Definition A.4 (Tensor Multiplication)**.** The outer product of two tensors, denoted as $\otimes$, is an operation that combines a pair of tensors to produce a higher-order tensor. Specifically, the outer product of two tensors $\mathbf{s}$ and $\mathbf{t}$, of orders p and q respectively, results in a tensor of order $p + q$, denoted as $\mathbf{o} = \mathbf{s} \otimes \mathbf{t}$, such that:

$$o_{i_1 \cdots i_p j_1 \dots j_q} = s_{i_1 \cdots i_p} t_{j_1 \cdots j_q}. \tag{24}$$

When the two tensors have the same dimension, the outer product is defined as $\otimes : \mathbb{R}^{n \circ p} \times \mathbb{R}^{n \circ q} \to \mathbb{R}^{n \circ (p+q)}$. For simplicity, we use the following shorthand notation:

$$\underbrace{a \otimes \cdots \otimes a}_{l \text{ times}} := a^{\otimes l}. \tag{25}$$

We also define the inner product between two tensors. The mode-l inner product between two tensors, assuming they share the same l-th dimension, is denoted as $\langle \mathbf{s}, \mathbf{t} \rangle_l$. Without loss of generality, assuming $l = 1$, the mode-l inner product is defined as:

$$[\langle \mathbf{s}, \mathbf{t} \rangle_l]_{i_2 \cdots i_p j_2 \cdots j_q} = \sum_{\iota=1}^{n_l} s_{\iota i_2 \cdots i_p} t_{\iota j_2 \cdots j_q}. \tag{26}$$

Note that in $\langle \cdot, \cdot \rangle_l$, the summation is performed over the l-th dimension of the first tensor, where the total number of elements is n_l. This definition can be naturally extended to multi-mode inner products by summing over multiple modes, written as $\langle \mathbf{a}, \mathbf{b} \rangle_{l_1, \cdots, l_m}$. We also refer to the tensor inner product as contraction, which involves contracting over m modes $l_1, \cdots, l_m$.

Lemma A.5 (Tensor Outer Product Identity)**.** *For four arbitrary matrices P, Q, U, V of compatible dimensions, the following identity holds:*

$$\langle P \otimes Q, U \otimes V \rangle_{2,4} = PU \otimes QV \tag{27}$$

Proof. See Section 10.2 of [30] for details. Note that this result corresponds to the tensor outer product version, where the tensor outer product is equivalent to the unreshaped version of the Kronecker product. ☐

Definition A.6 (Tensor Norm)**.** We adopt a definition similar to that in [31]. For a cubic tensor $\mathbf{s} \in \mathbb{R}^{nol}$, the spectral norm $\|\cdot\|_S$ and the nuclear norm $\|\cdot\|_*$ are defined as follows:

$$\|\mathbf{s}\|_* = \inf\left\{\sum_{j=1}^{r_m} |\lambda_j| : \mathbf{s} = \sum_{j=1}^{r_m} \lambda_j s_j^{\otimes l},\ \|s_j\|_2 = 1, s_j \in \mathbb{R}^n\right\}, \tag{28}$$

$$\|\mathbf{s}\|_S = \sup\left\{|\langle \mathbf{s}, a^{\otimes l}\rangle| : \|a\|_2 = 1, a \in \mathbb{R}^n\right\}. \tag{29}$$

Lemma A.7 (Dual Relationship Between Tensor Norms)**.** *The spectral norm $\|\cdot\|_S$ is the dual norm of the nuclear norm $\|\cdot\|_*$, meaning that for any tensor* $\mathbf{s}$*, the following holds:*

$$\|\mathbf{s}\|_S = \sup_{\|\mathbf{t}\|_* \le 1} |\langle \mathbf{s}, \mathbf{t}\rangle|, \tag{30}$$

where $\mathbf{t}$ is a tensor of the same dimensions as $\mathbf{s}$.

Proof. See Lemma 21 in [32]. ☐

Definition A.8 (Variational Eigenvalue of Tensor)**.** There are various ways to define eigenvalues of a tensor [33]. In this work, we adopt the same definition as in [21]. For a given tensor $\mathbf{s} \in \mathbb{R}^{nol}$, the k^{th} variational eigenvalue (v-Eigenvalue), denoted as $\lambda_k^v(\mathbf{s})$, is defined as:

$$\lambda_k^v(\mathbf{s}) := \max_{\substack{S \\ \dim(S)=k}} \min_{\mathbf{t} \in S} \frac{|\langle \mathbf{s}, \mathbf{t}\rangle|}{\|\mathbf{t}\|_F^2}, \quad k \in [n], \tag{31}$$

where S is a subspace of $\mathbb{R}^{nol}$ spanned by a set of orthogonal, symmetric, rank-1 tensors. The dimension of S corresponds to the number of orthogonal tensors that span this space. It follows directly from the definition that the spectral norm of $\mathbf{s}$ satisfies $\|\mathbf{s}\|_S = \lambda_1^v(\mathbf{s})$.

Theorem A.9 ($\mathbf{w}_t$ Tends to Rank-1, Adapted from [21])**.** *Consider the optimization problem* (6) *and its gradient descent (GD) trajectory over a finite horizon T, denoted as $\{\mathbf{w}_t\}_{t=0}^T$, with the update rule:*

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla h^l(\mathbf{w}_t), \tag{32}$$

where η is the stepsize. Then, there exist $t(\kappa, l) \ge 1$ and $\kappa < 1$ such that:

$$\frac{\lambda_2^v(\mathbf{w}_t)}{\lambda_1^v(\mathbf{w}_t)} \le \kappa, \qquad \forall t \in [t(\kappa, l), t_T], \tag{33}$$

provided that the initialization $\mathbf{w}_0$ is $\mathbf{w}_0 = \epsilon x_0^{\otimes l}$ with a sufficiently small ϵ. Here, $t(\kappa, l)$ is given by:

$$t(\kappa, l) = \left\lceil \ln\left(\frac{\|x_0\|_2^l}{\kappa |v_1^\top x_0|^l}\right) \ln\left(\frac{1 + \eta\sigma_1^l(U)}{1 + \eta\sigma_2^l(U)}\right)^{-1} \right\rceil \tag{34}$$

As long as

$$t \asymp \ln(1/\kappa) \cdot \ln\left(\frac{1 + \eta\sigma_1^l(U)}{1 + \eta\sigma_2^l(U)}\right)^{-1}, \tag{35}$$

the tensor $\mathbf{w}_t$ will be κ-rank-1, provided ϵ is selected as a function of U, r, n, L_s, and κ. Here, $U = \langle \mathbf{A}_r^* \mathbf{A}, M^* \rangle$, and $\asymp$ denotes "asymptotic to," meaning that the two terms on both sides of this symbol have the same order of magnitude.

We say a tensor $\mathbf{w}$ is κ-rank-1 if:

$$\lambda_2^v(\mathbf{w}) / \lambda_1^v(\mathbf{w}) \le \kappa. \tag{36}$$

Proof. See Theorem 1 in [21]. ☐

A.2 Matrix Sensing Basics

A.2.1 Restricted Strong Smoothness and Restricted Strong Convexity

Definition A.10 (Restricted Strong Smoothness (RSS) and Restricted Strong Convexity (RSC))**.** The Restricted Strong Smoothness (RSS) and Restricted Strong Convexity (RSC) properties [20] characterize the smoothness and curvature of a measurement operator $\mathcal{A}$, thereby providing a more flexible and expressive alternative to the Restricted Isometry Property (RIP) [11].

Specifically, a linear operator $\mathcal{A}$ satisfies the (L_s, p)-RSS and (α_s, p)-RSC properties if, for all $M, N \in \mathbb{R}^{n\times n}$ with $\operatorname{rank}(M), \operatorname{rank}(N) \leq p$, the following inequality holds:

$$\langle M - N, \nabla f(N)\rangle + \frac{\alpha_s}{2}\|M-N\|_F^2 \leq f(M) - f(N) \leq \langle M - N, \nabla f(N)\rangle + \frac{L_s}{2}\|M-N\|_F^2, \tag{37}$$

where $\nabla f(N)$ denotes the gradient of f at N, and f is the objective function defined in Equation (3).

Furthermore, the RIP constant δ_p can be expressed in terms of L_s and α_s as:

$$\delta_p = \frac{L_s - \alpha_s}{L_s + \alpha_s}. \tag{38}$$

This relationship reveals that the RSS and RSC properties generalize the RIP by decoupling the symmetric bounds of RIP into separate components, enabling a more granular analysis of the measurement operator's behavior.

A.2.2 First-Order and Second-Order Derivatives of the Matrix Space Objective Function

Lemma A.11 (Gradient and Hessian of the Matrix Objective Function)**.** *Let f be the matrix space objective function for the matrix sensing problem, as defined in Equation* (3)*, where $X \in \mathbb{R}^{n\times r_{search}}$. The gradient and Hessian of $f(XX^\top)$ are given as follows:*

Gradient:

$$\nabla f(XX^\top) = \sum_{i=1}^{m}\langle A_i, XX^\top - M^*\rangle A_i, \tag{39}$$

where $\nabla f(XX^\top)$ is a symmetric matrix because A_i are symmetric matrices.

Hessian:

$$\nabla^2 f(XX^\top)(XU^\top + UX^\top, XU^\top + UX^\top) = \sum_{i=1}^{m}\langle A_i, UX^\top + XU^\top\rangle^2. \tag{40}$$

Here, $A_i \in \mathbb{R}^{n\times n}$ are the sensing matrices, $M^* \in \mathbb{R}^{n\times n}$ is the ground truth matrix, and $U \in \mathbb{R}^{n\times r_{search}}$ is an arbitrary matrix.

Proof. The proof follows directly from standard multivariate calculus and properties of matrix optimization. For detailed derivations, refer to [34, 35, 36]. □

A.2.3 First-Order and Second-Order Derivatives of the Tensor Space Objective Function

Lemma A.12 (Gradient and Hessian of the Tensor Objective Function)**.** *Let f^l and h^l denote the tensor space objective functions for the matrix sensing problem, as defined in Equation* (7) *and Equation* (8)*, where $\mathbf{w} \in \mathbb{R}^{nr\circ l}$. The gradient and Hessian of $h^l(\mathbf{w})$ are given as follows:*

Gradient:

$$\nabla f^l(\mathbf{M}) = \left\langle\left\langle \mathbf{A}^{\otimes l}, \mathbf{M} - \mathcal{M}\left(\operatorname{vec}(Z)^{\otimes l}\right)\right\rangle, \mathbf{A}^{\otimes l}\right\rangle_{1,4,\cdots,3l-2}, \tag{41}$$

$$\nabla h^l(\mathbf{X}) = 2\left\langle \nabla f^l\left(\langle \mathbf{X}, \mathbf{X}\rangle_{2*[l]}\right), \mathbf{X}\right\rangle_{2*[l]}, \tag{42}$$

$$\nabla_{\mathbf{w}} h^l(\mathbf{P}(\mathbf{w})) = \left\langle \nabla h^l(\mathbf{X}), \mathbf{P}^{\otimes l}\right\rangle_{1,2,4,5,\cdots,3l-1,3l}, \tag{43}$$

where $\mathbf{M} \in \mathbb{R}^{n\circ 2l}$ and $\mathbf{X} \in \mathbb{R}^{[n\times r]\circ l}$. $ZZ^\top = M^*$ is the ground truth matrix in the matrix sensing problem. The map $\mathcal{M} : \mathbb{R}^{nr\circ l} \to \mathbb{R}^{n\circ 2l}$ is defined as:

$$\mathcal{M}(\mathbf{w}) = \langle \mathbf{P}(\mathbf{w}), \mathbf{P}(\mathbf{w})\rangle_{2*[l]}, \tag{44}$$

and its total derivative at $\mathbf{w}$ *is the linear map* $\mathcal{D}_{\mathbf{w}}\mathcal{M} : \mathbb{R}^{nrol} \to \mathbb{R}^{no2l}$ *given by:*

$$\mathcal{D}_{\mathbf{w}}\mathcal{M}(\mathbf{v}) = \langle \mathbf{P}(\mathbf{v}), \mathbf{P}(\mathbf{w})\rangle_{2*[l]} + \langle \mathbf{P}(\mathbf{w}), \mathbf{P}(\mathbf{v})\rangle_{2*[l]} \,. \tag{45}$$

Hessian:

$$\left[\nabla^2_{\mathbf{w}} h^l\left(\mathbf{P}(\mathbf{w})\right)\right](\mathbf{v},\mathbf{v}) = 2\left\langle \nabla f^l(\mathcal{M}(\mathbf{w})), \mathcal{M}(\mathbf{v})\right\rangle + \left\|\left\langle \mathbf{A}^{\otimes l}, \mathcal{D}_{\mathbf{w}}\mathcal{M}(\mathbf{v})\right\rangle\right\|_F^2 . \tag{46}$$

Here, $\mathbf{A}^{\otimes l}$ *represents a tensor formed by the tensor outer product of* $\mathbf{A} \in \mathbb{R}^{m\times n\times n}$ *with itself* l *times, where* $\mathbf{A}$ *consists of* m *sensing matrices.*

Proof. The proof follows from the results in [10], specifically Lemma 5.2 Detailed derivations can be found therein. □

B Proofs of Main Theorems

B.1 Proof of Theorem 3.2

Proof. If the step size η is sufficiently small, we can use the Taylor expansion of Equation (11) around $\hat{\mathbf{w}}$:

$$\begin{aligned} h^l(\hat{\mathbf{w}} + \eta\,\mathrm{vec}(u_n q_r^\top)^{\otimes l}) &\approx h^l(\hat{\mathbf{w}}) + \eta\langle \nabla h^l(\hat{\mathbf{w}}), \mathrm{vec}(u_n q_r^\top)^{\otimes l}\rangle + \frac{\eta^2}{2}\left[\nabla^2 h^l(\hat{\mathbf{w}})\right]\left(\mathrm{vec}(u_n q_r^\top)^{\otimes l}, \mathrm{vec}(u_n q_r^\top)^{\otimes l}\right) \\ &:= h^l(\hat{\mathbf{w}}) + \eta c_1 + \frac{\eta^2}{2} c_2 \end{aligned} \tag{47}$$

where c_1 and c_2 are the linear and quadratic terms, respectively.

According to Theorem 5.3 in [10], if $\hat{X}$ is a first-order critical point of Equation (3), and $\hat{\mathbf{w}} = \mathrm{vec}(\hat{X})^{\otimes l}$, then:

$$\nabla h^l(\hat{\mathbf{w}}) = 0. \tag{48}$$

This implies that $c_1 = 0$.

Furthermore, by the proof of Theorem 3.1, which corresponds to Theorem 5.4 in [10], the quadratic term c_2 is always negative. Substituting these results into the Taylor expansion, we have:

$$h^l(\hat{\mathbf{w}} + \eta\,\mathrm{vec}(u_n q_r^\top)^{\otimes l}) \approx h^l(\hat{\mathbf{w}}) + \frac{\eta^2}{2} c_2, \tag{49}$$

where $\frac{\eta^2}{2} c_2 < 0$ for any $\eta > 0$.

Therefore, it is evident that:

$$h^l(\hat{\mathbf{w}} + \eta\,\mathrm{vec}(u_n q_r^\top)^{\otimes l}) < h^l(\hat{\mathbf{w}}), \tag{50}$$

which completes the proof. □

B.2 Proof of Theorem 4.1

Before presenting the proof of Theorem 4.1, we establish two key lemmas in the following subsubsections. These lemmas lay the foundation for the main result. The proof of the theorem itself is provided in the third subsubsection.

B.2.1 Matrix Space Trajectory Decomposition and Approximation

The first lemma concerns the decomposition and approximation of the trajectory in the matrix space under gradient descent (GD).

Lemma B.1. *If the initialization is given by* $X_0 = \epsilon X \in \mathbb{R}^{n\times r_{\text{search}}}$ *with* $|X|_F^2 = 1$*, then the trajectory of* X_t *can be approximated as:*

$$X_t \approx \tilde{X}_t = \left(I + \eta\sum_{i=1}^{m}\langle A_i, M^*\rangle A_i\right)^t X_0 \tag{51}$$

Proof. We begin by analyzing the first step of the gradient descent (GD) update:

$$\begin{aligned} X_1 = X_0 - \eta \nabla h\left(X_0\right) &= X_0 - \eta \left(\sum_{i=1}^{m} \left\langle A_i, X_0 X_0^\top - M^* \right\rangle A_i \right) X_0 \\ &= \left(I + \eta \sum_{i=1}^{m} \left\langle A_i, M^* \right\rangle A_i \right) X_0 - \eta \left(\sum_{i=1}^{m} \left\langle A_i, X_0 X_0^\top \right\rangle A_i \right) X_0. \end{aligned} \tag{52}$$

The first term in Equation (52) captures the leading-order update due to the initialization, while the second term is of higher order in ϵ. Specifically, since $X_0 = \epsilon X$, we can expand the second term as:

$$-\eta \left(\sum_{i=1}^{m} \left\langle A_i, X_0 X_0^\top \right\rangle A_i \right) X_0 = \mathcal{O}(\epsilon^3). \tag{53}$$

Thus, the update simplifies to:

$$X_1 = \left(I + \eta \sum_{i=1}^{m} \left\langle A_i, M^* \right\rangle A_i \right) X_0 + \mathcal{O}(\epsilon^3). \tag{54}$$

By induction, this result can be generalized to t steps under the assumption that $\epsilon \to 0$, such that the higher-order terms $\mathcal{O}(\epsilon^3)$ vanish. Consequently, the trajectory of X_t can be approximated as shown in Equation (51).

This completes the proof of the lemma. □

B.2.2 Tensor Space Trajectory Decomposition and Approximation

The second lemma addresses both the decomposition and approximation of the trajectory in the tensor space under gradient descent (GD).

Lemma B.2. *For any point* $\mathbf{w}_t$ *on the GD trajectory, the following decomposition holds:*

$$\mathbf{w}_t = \langle \mathbf{z}_t, \mathbf{w}_0 \rangle - \mathbf{E}_t = \tilde{\mathbf{w}}_t - \mathbf{E}_t, \tag{55}$$

where:

$$\mathbf{z}_t = \left(\mathcal{J} + \eta \left\langle \left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle, M^{*\otimes l} \right\rangle \right)^t \in \mathbb{R}^{[nr \times nr] \circ l}, \tag{56}$$

$$\mathbf{E}_t = \sum_{i=1}^{t} \left\langle \left(\mathcal{J} + \eta \left\langle \left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle, M^{*\otimes l} \right\rangle \right)^{t-i}, \hat{\mathbf{E}}_i \right\rangle, \tag{57}$$

$$\hat{\mathbf{E}}_i = \eta \left\langle \left\langle \left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle, \left\langle \mathbf{P}\left(\mathbf{w}_{i-1}\right), \mathbf{P}\left(\mathbf{w}_{i-1}\right) \right\rangle_{2*[l]} \right\rangle, \mathbf{w}_{i-1} \right\rangle_{2*[l]} \in \mathbb{R}^{nr \circ l}. \tag{58}$$

Here, $\mathbf{A}_r = I_r \oslash_{1,2} \mathbf{A} \in \mathbb{R}^{rm \times rn \times n}$ *represents the Kronecker product applied only to the first and second dimensions of* $\mathbf{A}$*, making* $\left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle \in \mathbb{R}^{[rn \times rn \times n \times n] \circ l}$*. Additionally,* $\mathcal{J} \in \mathbb{R}^{[rn \times rn] \circ l}$ *denotes the identity operator.*

Furthermore, if $\mathbf{w}_0 = \epsilon \mathbf{w} \in \mathbb{R}^{nr \circ l}$*, then* $\|\mathbf{E}_t\|_S = \mathcal{O}(\epsilon^3)$*. Under the assumption that* ϵ *is sufficiently small, this implies:*

$$\mathbf{w}_t \approx \langle \mathbf{z}_t, \mathbf{w}_0 \rangle. \tag{59}$$

Proof. The proof follows the same steps as the proofs of Lemmas 12 and 13 in [21]. For brevity, the details are omitted here. □

B.2.3 Approximation Relationship Between Matrix Space and Tensor Space Trajectories

In this subsection, we present the detailed proof of Theorem 4.1. In the process, we utilize the delta function δ_{ij}, which takes the value 1 if $i = j$ and 0 otherwise.

Proof. First, we write the value of each component of a point on the approximate tensor gradient descent (GD) trajectory:

$$\begin{aligned}
\tilde{\mathbf{w}}_{t j_1 p_1 \cdots j_l p_l} &= \sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n} \sum_{q_1=1,\cdots,q_l=1}^{r,\cdots,r} \left(\mathcal{J} + \eta \left\langle \left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle, M^{*\otimes l} \right\rangle \right)^t_{j_1 p_1 k_1 q_1 \cdots j_l p_l k_l q_l} \mathbf{w}_{0 k_1 q_1 \cdots k_l q_l} \\
&= \sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n} \sum_{q_1=1,\cdots,q_l=1}^{r,\cdots,r} \left(\mathcal{J}_{j_1 p_1 k_1 q_1 \cdots j_l p_l k_l q_l} + \eta \left\langle \left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle, M^{*\otimes l} \right\rangle_{j_1 p_1 k_1 q_1 \cdots j_l p_l k_l q_l} \right)^t \mathbf{w}_{0 k_1 q_1 \cdots k_l q_l} \\
&= \sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n} \sum_{q_1=1,\cdots,q_l=1}^{r,\cdots,r} \left(\mathcal{J}_{j_1 p_1 k_1 q_1 \cdots j_l p_l k_l q_l} + \eta \left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle_{p_1 q_1 \cdots p_l q_l} M^{*\otimes l}_{j_1 k_1 \cdots j_l k_l} \right)^t \mathbf{w}_{0 k_1 q_1 \cdots k_l q_l}. \qquad (60)
\end{aligned}$$

Next, the term $\left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle_{p_1 q_1 \cdots p_l q_l} M^{*\otimes l}_{j_1 k_1 \cdots j_l k_l}$ can be expressed as:

$$\begin{aligned}
&\left\langle \mathbf{A}_r^{\otimes l}, \mathbf{A}^{\otimes l} \right\rangle_{p_1 q_1 \cdots p_l q_l} M^{*\otimes l}_{j_1 k_1 \cdots j_l k_l} \\
&= \sum_{i_1=1,\cdots,i_l=1}^{m,\cdots,m} \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n} \mathbf{A}_{r p_1 i_1, q_1 j_1, k_1} \cdots \mathbf{A}_{r p_l i_l, q_l j_l, k_l} \mathbf{A}_{i_1 j_1 k_1} \cdots \mathbf{A}_{i_l j_l k_l} M^*_{j_1 k_1} \cdots M^*_{j_l k_l} \\
&= \sum_{i_1=1,\cdots,i_l=1}^{m,\cdots,m} \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n} I_{r p_1 q_1} \mathbf{A}_{i_1 j_1 k_1} \mathbf{A}_{i_1 j_1 k_1} M^*_{j_1 k_1} \cdots I_{r p_l q_l} \mathbf{A}_{i_l j_l k_l} \mathbf{A}_{i_l j_l k_l} M^*_{j_l k_l} \\
&= \sum_{i_1=1,\cdots,i_l=1}^{m,\cdots,m} I_{r p_1 q_1} \left\langle \mathbf{A}_{i_1,:,:}, M^* \right\rangle \mathbf{A}_{i_1 j_1 k_1} \cdots I_{r p_l q_l} \left\langle \mathbf{A}_{i_l,:,:}, M^* \right\rangle \mathbf{A}_{i_l j_l k_l} \\
&= \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \sum_{i_1=1,\cdots,i_l=1}^{m,\cdots,m} \left\langle \mathbf{A}_{i_1,:,:}, M^* \right\rangle \mathbf{A}_{i_1 j_1 k_1} \cdots \left\langle \mathbf{A}_{i_l,:,:}, M^* \right\rangle \mathbf{A}_{i_l j_l k_l} \\
&= \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \sum_{i_1=1}^{m} \cdots \sum_{i_l=1}^{m} \left[\left\langle \mathbf{A}_{i_1,:,:}, M^* \right\rangle \mathbf{A}_{i_1,:,:}\right]_{j_1 k_1} \cdots \left[\left\langle \mathbf{A}_{i_l,:,:}, M^* \right\rangle \mathbf{A}_{i_l,:,:}\right]_{j_l k_l} \\
&= \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \left[\sum_{i_1=1}^{m} \left\langle A_{i_1}, M^* \right\rangle A_{i_1}\right]_{j_1 k_1} \cdots \left[\sum_{i_l=1}^{m} \left\langle A_{i_l}, M^* \right\rangle A_{i_l}\right]_{j_l k_l}. \qquad (61)
\end{aligned}$$

Substituting Equation (61) into Equation (60), we obtain:

$$\begin{aligned}
\tilde{\mathbf{w}}_{t j_1 p_1 \cdots j_l p_l} = \sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n} \sum_{q_1=1,\cdots,q_l=1}^{r,\cdots,r} &(\delta_{j_1 k_1} \delta_{p_1 q_1} \cdots \delta_{j_1 k_1} \delta_{p_l q_l} + \\
&\eta \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \left[\sum_{i_1=1}^{m} \left\langle A_{i_1}, M^* \right\rangle A_{i_1}\right]_{j_1 k_1} \cdots \left[\sum_{i_l=1}^{m} \left\langle A_{i_l}, M^* \right\rangle A_{i_l}\right]_{j_l k_l})^t \cdot \mathbf{w}_{0 k_1 q_1 \cdots k_l q_l} \qquad (62)
\end{aligned}$$

In Equation (62), each term in $\sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n} \sum_{q_1=1,\cdots,q_l=1}^{r,\cdots,r}$ can be derived as:

$$\begin{aligned}
&\left(\delta_{j_1 k_1} \delta_{p_1 q_1} \cdots \delta_{j_1 k_1} \delta_{p_l q_l} + \eta \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \left[\sum_{i_1=1}^{m} \left\langle A_{i_1}, M^* \right\rangle A_{i_1}\right]_{j_1 k_1} \cdots \left[\sum_{i_l=1}^{m} \left\langle A_{i_l}, M^* \right\rangle A_{i_l}\right]_{j_l k_l} \right)^t \mathbf{w}_{0 k_1 q_1 \cdots k_l q_l} \\
&= \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \left[\left(I + \eta^{\frac{1}{l}} \sum_{i=1}^{m} \left\langle A_i, M^* \right\rangle A_i\right)^t\right]^{\otimes l}_{j_1 k_1 \cdots j_l k_l} \mathbf{w}_{0 k_1 q_1 \cdots k_l q_l} \\
&= \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \left[\left(I + \eta^{\frac{1}{l}} \sum_{i=1}^{m} \left\langle A_i, M^* \right\rangle A_i\right)^t\right]^{\otimes l}_{j_1 k_1 \cdots j_l k_l} X_{0 k_1 q_1} \cdots X_{0 k_l q_l} \\
&= \delta_{p_1 q_1} \cdots \delta_{p_l q_l} \left[\left(I + \eta^{\frac{1}{l}} \sum_{i=1}^{m} \left\langle A_i, M^* \right\rangle A_i\right)^t\right]_{j_1 k_1} X_{0 k_1 q_1} \cdots \left[\left(I + \eta^{\frac{1}{l}} \sum_{i=1}^{m} \left\langle A_i, M^* \right\rangle A_i\right)^t\right]_{j_l k_l} X_{0 k_l q_l} \qquad (63)
\end{aligned}$$

Under the operation of $\sum_{k_1=1,\cdots,k_l=1}^{n,\cdots,n}$, we obtained l-fold matrix multiplications. The resulting expression can be further derived under the operation of $\sum_{q_1=1,\cdots,q_l=1}^{r,\cdots,r}$:

$$\begin{aligned}
&\sum_{q_1=1,\cdots,q_l=1}^{r,\cdots,r} \delta_{p_1q_1}\cdots\delta_{p_lq_l}\left[\left(I+\eta^{\frac{1}{l}}\sum_{i=1}^{m}\langle A_i,M^*\rangle A_i\right)^t X_0\right]_{j_1q_1}\cdots\left[\left(I+\eta^{\frac{1}{l}}\sum_{i=1}^{m}\langle A_i,M^*\rangle A_i\right)^t X_0\right]_{j_lq_l} \\
&=\sum_{q_1=1}^{r}\delta_{p_1q_1}\left[\left(I+\eta^{\frac{1}{l}}\sum_{i=1}^{m}\langle A_i,M^*\rangle A_i\right)^t X_0\right]_{j_1q_1}\cdots\sum_{q_l=1}^{r}\delta_{p_lq_l}\left[\left(I+\eta^{\frac{1}{l}}\sum_{i=1}^{m}\langle A_i,M^*\rangle A_i\right)^t X_0\right]_{j_lq_l} \\
&=\left[\left(I+\eta^{\frac{1}{l}}\sum_{i=1}^{m}\langle A_i,M^*\rangle A_i\right)^t X_0\right]_{j_1p_1}\cdots\left[\left(I+\eta^{\frac{1}{l}}\sum_{i=1}^{m}\langle A_i,M^*\rangle A_i\right)^t X_0\right]_{j_lp_l} \\
&=\tilde{X}_{tj_1p_1}\cdots\tilde{X}_{tj_lp_l}=\operatorname{vec}(\tilde{X}_t)^{\otimes l}_{j_1p_1,\cdots,j_lp_l}
\end{aligned} \tag{64}$$

This demonstrates that, when approximating the gradient descent (GD) trajectory at any time step t, the trajectory in the tensor space has a component $\tilde{\mathbf{w}}_t$ that corresponds one-to-one with the component $\tilde{X}_t$ of the trajectory in the matrix space. The key difference is that the effective step size in the tensor space is η, while in the matrix space, it is scaled by $\eta^{\frac{1}{l}}$.

This completes the proof. □

B.3 Proof of Theorem 4.3

This subsection is dedicated to proving one of the most significant theorems in our work: Theorem 4.3. The proof is structured as follows. In the first subsubsection, we introduce an important lemma that serves as the foundation for the proof. Building on this lemma, we propose a novel and intuitive approach, which we term the "dumpling" method, to complete the proof. Specifically, in the second and third subsubsections, we demonstrate that $\check{\mathbf{w}}$ and $\hat{\mathbf{w}}+\eta\operatorname{vec}(u_nq_r^\top)^{\otimes l}$ can each be approximated in the directions of $\operatorname{vec}(u_nq_r^\top)^{\otimes l}$ and $\operatorname{vec}(v_rq_r^\top)^{\otimes l}$, respectively. This step is akin to aligning the two sides of a dumpling wrapper before sealing it. Finally, in the fourth subsubsection, we establish that these two components can be approximated within the unit tensor subspace spanned by $\operatorname{vec}(u_nq_r^\top)^{\otimes l}$ and $\operatorname{vec}(v_rq_r^\top)^{\otimes l}$. This step completes the proof by showing that the approximation aligns with the tensor spectral norm, akin to neatly sealing the remaining edges of the dumpling wrapper to ensure a perfect fit.

B.3.1 Orthogonality Between Singular Vector v_r and Eigenvector u_n

Lemma B.3. *If $\hat{X}$ is a first-order critical point of Equation* (3) *and the smallest eigenvalue of $\nabla f(\hat{X}\hat{X}^\top)$ is negative, then the left singular vector v_r of $\hat{X}$ corresponding to its smallest singular value is orthogonal to the eigenvector u_n of $\nabla f(\hat{X}\hat{X}^\top)$ corresponding to its smallest eigenvalue.*

Proof. We first factorize the point $\hat{X}$ and the gradient $\nabla f(\hat{X}\hat{X}^\top)$ using singular value decomposition (SVD) and eigenvalue decomposition (EVD), respectively:

$$\hat{X}=\sum_{\phi=1}^{r}\sigma_\phi v_\phi q_\phi^\top \tag{65}$$

$$\nabla f(\hat{X}\hat{X}^\top)=\sum_{\theta=1}^{n}\lambda_\theta u_\theta u_\theta^\top \tag{66}$$

where σ_ϕ are the singular values of $\hat{X}$, v_ϕ and q_ϕ are the left and right singular vectors, λ_θ are the eigenvalues of $\nabla f(\hat{X}\hat{X}^\top)$, and u_θ are the corresponding eigenvectors.

Since $\hat{X}$ is a first-order critical point of $f(\hat{X}\hat{X}^\top)$ and $\nabla f(\hat{X}\hat{X}^\top)$ is a symmetric matrix, it satisfies:

$$\nabla f(\hat{X}\hat{X}^\top)v_r=\nabla f(\hat{X}\hat{X}^\top)^\top v_r=\left(\sum_{\theta=1}^{n}\lambda_\theta u_\theta u_\theta^\top\right)v_r=0, \tag{67}$$

where v_r is the left singular vector of $\hat{X}$ corresponding to its smallest singular value σ_r.

Next, we expand v_r in terms of the eigenvector basis u_θ of $\nabla f(\hat{X}\hat{X}^\top)$:

$$v_r = \sum_{\theta=1}^{n} c_\theta u_\theta, \quad \text{where} \quad c_\theta = u_\theta^\top v_r. \tag{68}$$

Substituting this expansion into $\nabla f(\hat{X}\hat{X}^\top)v_r = 0$, we get:

$$\nabla f(\hat{X}\hat{X}^\top)v_r = \sum_{\theta=1}^{n} \lambda_\theta c_\theta u_\theta = 0. \tag{69}$$

Since the eigenvectors u_θ are linearly independent, the coefficients must satisfy:

$$\lambda_\theta c_\theta = 0, \quad \forall \theta. \tag{70}$$

This implies:

$$c_\theta = 0 \quad \text{for all} \quad \theta \quad \text{such that} \quad \lambda_\theta \neq 0. \tag{71}$$

Thus, v_r must lie entirely in the eigenspace corresponding to the zero eigenvalue of $\nabla f(\hat{X}\hat{X}^\top)$. Let the set of eigenvectors corresponding to the zero eigenvalue be u_{θ_0}. Then:

$$v_r \in \operatorname{span}\left(\{u_{\theta_0}\}\right). \tag{72}$$

Since $\lambda_n < 0$, the eigenvector u_n corresponds to a strictly negative eigenvalue. Therefore, u_n cannot belong to the null space of $\nabla f(\hat{X}\hat{X}^\top)$.

Because eigenvectors corresponding to distinct eigenvalues of a symmetric matrix are orthogonal, the eigenspace of the zero eigenvalue is orthogonal to the eigenspace of the negative eigenvalue λ_n. Hence:

$$u_n^\top v_r = 0. \tag{73}$$

This completes the proof. □

Corollary B.4 (Orthogonality Between Tensor Space Projection Directions)**.** *Based on the above lemma, we immediately obtain the following result:*

$$\operatorname{vec}(u_n q_r^\top)^{\otimes l} \perp \operatorname{vec}(v_r q_r^\top)^{\otimes l} \tag{74}$$

Proof. The proof is omitted for brevity. □

B.3.2 Approximation Relationship in the Projection Direction of $\operatorname{vec}(u_n q_r^\top)^{\otimes l}$

Lemma B.5. *The condition*

$$\sigma_r^l \alpha^{\frac{l}{2}} \to \eta \tag{75}$$

is necessary and sufficient for $\check{\mathbf{w}}$ *to approximate* $\hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}$ *in the projection direction of* $\operatorname{vec}(u_n q_r^\top)^{\otimes l}$.

Proof. To establish this result, we consider the explicit form of the approximation condition:

$$\begin{aligned} &\left\langle \check{\mathbf{w}}^{\otimes l}, \operatorname{vec}(u_n q_r^\top)^{\otimes l} \right\rangle \to \left\langle \hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}, \operatorname{vec}(u_n q_r^\top)^{\otimes l} \right\rangle \\ \Leftrightarrow &\left\langle \left[\sum_{\phi=1}^{r-1} \sigma_\phi v_\phi q_\phi^\top + \sigma_r(\sqrt{\alpha} u_n + \sqrt{\beta} v_r) q_r^\top \right]^{\otimes l}, (u_n q_r^\top)^{\otimes l} \right\rangle \to \left\langle \hat{X}^{\otimes l} + \eta (u_n q_r^T)^{\otimes l}, (u_n q_r^T)^{\otimes l} \right\rangle \end{aligned} \tag{76}$$

To verify that Equation (76) holds, we expand the inner product term into its component elements:

$$\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left[\sum_{\phi=1}^{r-1}\sigma_\phi v_\phi q_\phi^\top + \sigma_r(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)q_r^\top\right]^{\otimes l}_{j_1p_1\cdots j_lp_l} (u_nq_r^\top)^{\otimes l}_{j_1p_1\cdots j_lp_l}$$

$$\to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left[\hat{X}^{\otimes l} + \eta(u_nq_r^\top)^{\otimes l}\right]_{j_1p_1\cdots j_lp_l} (u_nq_r^\top)^{\otimes l}_{j_1p_1\cdots j_lp_l}$$

$$\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l}\left[\sum_{\phi=1}^{r-1}\sigma_\phi v_{\phi,j_\iota}q_{\phi,p_\iota} + \sigma_r(\sqrt{\alpha}u_{n,j_\iota} + \sqrt{\beta}v_{r,j_\iota})q_{r,p_\iota}\right]\prod_{\iota=1}^{l}u_{n,j_\iota}q_{r,p_\iota}$$

$$\to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left[\prod_{\iota=1}^{l}\left(\sum_{\phi=1}^{r-1}\sigma_\phi v_{\phi,j_\iota}q_{\phi,p_\iota} + \sigma_r v_{r,j_\iota}q_{r,p_\iota}\right) + \eta\prod_{\iota=1}^{l}u_{n,j_\iota}q_{r,p_\iota}\right]\prod_{\iota=1}^{l}u_{n,j_\iota}q_{r,p_\iota}$$

$$\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l}\left[\sum_{\phi=1}^{r-1}\sigma_\phi v_{\phi,j_\iota}q_{\phi,p_\iota} + \sigma_r(\sqrt{\alpha}u_{n,j_\iota} + \sqrt{\beta}v_{r,j_\iota})q_{r,p_\iota}\right]u_{n,j_\iota}q_{r,p_\iota}$$

$$\to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l}\left(\sum_{\phi=1}^{r-1}\sigma_\phi v_{\phi,j_\iota}q_{\phi,p_\iota} + \sigma_r v_{r,j_\iota}q_{r,p_\iota}\right)u_{n,j_\iota}q_{r,p_\iota} + \eta\prod_{\iota=1}^{l}u_{n,j_\iota}q_{r,p_\iota}u_{n,j_\iota}q_{r,p_\iota} \tag{77}$$

Using the orthogonality of the singular vectors $q_r \perp q_\phi$ (where $\phi = 1,\cdots,r-1$), derived from the Singular Value Decomposition (SVD), all terms involving $q_{\phi,p_\iota}q_{r,p_\iota}$ vanish under the summation $\sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r}$. Consequently, we have:

$$\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l}\sigma_r(\sqrt{\alpha}u_{n,j_\iota} + \sqrt{\beta}v_{r,j_\iota})q_{r,p_\iota}u_{n,j_\iota}q_{r,p_\iota}$$

$$\to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left(\prod_{\iota=1}^{l}\sigma_r v_{r,j_\iota}q_{r,p_\iota}u_{n,j_\iota}q_{r,p_\iota} + \eta\prod_{\iota=1}^{l}u_{n,j_\iota}q_{r,p_\iota}u_{n,j_\iota}q_{r,p_\iota}\right) \tag{78}$$

Furthermore, using the orthogonality $v_r \perp u_n$ (by Lemma B.3), all terms involving $v_{r,j_\iota}u_{n,j_\iota}$ vanish under the summation $\sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n}$. This reduces the expression to:

$$\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l}\sigma_r\sqrt{\alpha}u_{n,j_\iota}q_{r,p_\iota}u_{n,j_\iota}q_{r,p_\iota} \to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \eta\prod_{\iota=1}^{l}u_{n,j_\iota}q_{r,p_\iota}u_{n,j_\iota}q_{r,p_\iota}$$

$$\Leftrightarrow \sigma_r^l\alpha^{\frac{l}{2}} \to \eta \Leftrightarrow \alpha \to \left(\frac{\eta}{\sigma_r^l}\right)^{\frac{2}{l}} \tag{79}$$

This completes the proof. □

This establishes the necessary and sufficient condition for the approximation in the projection direction $\mathrm{vec}(u_nq_r^\top)^{\otimes l}$. At this stage, we have successfully aligned one side of the dumpling wrapper.

B.3.3 Approximation Relationship in the Projection Direction of $\mathrm{vec}(v_rq_r^\top)^{\otimes l}$

Lemma B.6. *The condition*

$$\beta \to 1 \tag{80}$$

is necessary and sufficient for $\check{\mathbf{w}}$ *to approximate* $\hat{\mathbf{w}} + \eta\,\mathrm{vec}(u_nq_r^\top)^{\otimes l}$ *in the projection direction of* $\mathrm{vec}(v_rq_r^\top)^{\otimes l}$.

Proof. To establish this result, we consider the explicit form of the approximation condition:

$$\left\langle \check{\mathbf{w}}^{\otimes l}, \mathrm{vec}(v_rq_r^\top)^{\otimes l}\right\rangle \to \left\langle \hat{\mathbf{w}} + \eta\,\mathrm{vec}(u_nq_r^\top)^{\otimes l}, \mathrm{vec}(v_rq_r^\top)^{\otimes l}\right\rangle$$

$$\Leftrightarrow \left\langle \left[\sum_{\phi=1}^{r-1}\sigma_\phi v_\phi q_\phi^\top + \sigma_r(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)q_r^\top\right]^{\otimes l}, (v_rq_r^\top)^{\otimes l}\right\rangle \to \left\langle \hat{X}^{\otimes l} + \eta(u_nq_r^T)^{\otimes l}, (v_rq_r^T)^{\otimes l}\right\rangle \tag{81}$$

To verify that Equation (81) holds, we expand the inner product term into its component elements:

$$
\begin{aligned}
&\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left[\sum_{\phi=1}^{r-1} \sigma_\phi v_\phi q_\phi^\top + \sigma_r(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)q_r^\top\right]^{\otimes l}_{j_1p_1\cdots j_lp_l} (v_r q_r^\top)^{\otimes l}_{j_1p_1\cdots j_lp_l} \\
&\quad \to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left[\hat{X}^{\otimes l} + \eta(u_n q_r^\top)^{\otimes l}\right]_{j_1p_1\cdots j_lp_l} (v_r q_r^\top)^{\otimes l}_{j_1p_1\cdots j_lp_l} \\
&\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l} \left[\sum_{\phi=1}^{r-1} \sigma_\phi v_{\phi,j_\iota} q_{\phi,p_\iota} + \sigma_r(\sqrt{\alpha}u_{n,j_\iota} + \sqrt{\beta}v_{r,j_\iota})q_{r,p_\iota}\right] \prod_{\iota=1}^{l} v_{r,j_\iota} q_{r,p_\iota} \\
&\quad \to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left[\prod_{\iota=1}^{l}\left(\sum_{\phi=1}^{r-1} \sigma_\phi v_{\phi,j_\iota} q_{\phi,p_\iota} + \sigma_r v_{r,j_\iota} q_{r,p_\iota}\right) + \eta \prod_{\iota=1}^{l} u_{n,j_\iota} q_{r,p_\iota}\right] \prod_{\iota=1}^{l} v_{r,j_\iota} q_{r,p_\iota} \\
&\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l} \left[\sum_{\phi=1}^{r-1} \sigma_\phi v_{\phi,j_\iota} q_{\phi,p_\iota} + \sigma_r(\sqrt{\alpha}u_{n,j_\iota} + \sqrt{\beta}v_{r,j_\iota})q_{r,p_\iota}\right] v_{r,j_\iota} q_{r,p_\iota} \\
&\quad \to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l}\left(\sum_{\phi=1}^{r-1} \sigma_\phi v_{\phi,j_\iota} q_{\phi,p_\iota} + \sigma_r v_{r,j_\iota} q_{r,p_\iota}\right) v_{r,j_\iota} q_{r,p_\iota} + \eta \prod_{\iota=1}^{l} u_{n,j_\iota} q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota}
\end{aligned} \tag{82}
$$

Using the orthogonality of the singular vectors $q_r \perp q_\phi$ (where $\phi = 1, \cdots, r-1$), derived from the Singular Value Decomposition (SVD), all terms involving $q_{\phi,p_\iota} q_{r,p_\iota}$ vanish under the summation $\sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r}$. Consequently, we have:

$$
\begin{aligned}
&\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l} \sigma_r(\sqrt{\alpha}u_{n,j_\iota} + \sqrt{\beta}v_{r,j_\iota})q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota} \\
&\quad \to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \left(\prod_{\iota=1}^{l} \sigma_r v_{r,j_\iota} q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota} + \eta \prod_{\iota=1}^{l} u_{n,j_\iota} q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota}\right)
\end{aligned} \tag{83}
$$

Furthermore, using the orthogonality $v_r \perp u_n$ (by Lemma B.3), all terms involving $u_{n,j_\iota} v_{r,j_\iota}$ vanish under the summation $\sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n}$. This reduces the expression to:

$$
\begin{aligned}
&\Leftrightarrow \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l} \sigma_r \sqrt{\beta} v_{r,j_\iota} q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota} \to \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l} \sigma_r v_{r,j_\iota} q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota} \\
&\Leftrightarrow \beta^{\frac{l}{2}} \to 1 \Leftrightarrow \beta \to 1
\end{aligned} \tag{84}
$$

This completes the proof. □

This establishes the necessary and sufficient condition for the approximation in the projection direction $\mathrm{vec}(v_r q_r^\top)^{\otimes l}$. At this stage, we have successfully aligned another side of the dumpling wrapper.

B.3.4 Approximation Guarantee in Terms of Tensor Norm

We now formally present the proof of Theorem 4.3.

Proof. We begin by defining the residual tensor $\mathbf{d}_p$ as follows:

$$\mathbf{d}_p := \check{\mathbf{w}} - \left[\hat{\mathbf{w}} + \eta\, \mathrm{vec}(u_n q_r^\top)^{\otimes l}\right]. \tag{85}$$

According to the definitions of the tensor spectral norm and nuclear norm, we have:

$$\left\|\mathrm{vec}(u_n q_r^\top)^{\otimes l}\right\|_* = \left\|\mathrm{vec}(v_r q_r^\top)^{\otimes l}\right\|_* = 1. \tag{86}$$

Here, u_n, v_r, and q_r are unit vectors, and hence their norms are all 1.

Next, we define $\mathbf{d}_s$ in terms of the spectral norm of $\mathbf{d}_p$, as established in Lemma A.7:

$$\|\mathbf{d}_p\|_S = |\langle \mathbf{d}_p, \mathbf{d}_s \rangle|, \quad \|\mathbf{d}_s\|_* \leq 1, \quad \mathbf{d}_s \in \mathbb{R}^{nrol}. \tag{87}$$

From Corollary B.4, we know that $\mathbf{d}_s$ must lie in the unit tensor space spanned by $\operatorname{vec}(u_n q_r^\top)^{\otimes l}$ and $\operatorname{vec}(v_r q_r^\top)^{\otimes l}$. Specifically, we have:

$$\mathbf{d}_s = \lambda_{u_n} \operatorname{vec}(u_n q_r^\top)^{\otimes l} + \lambda_{v_r} \operatorname{vec}(v_r q_r^\top)^{\otimes l} \Rightarrow |\lambda_{u_n}| + |\lambda_{v_r}| \leq 1. \tag{88}$$

Using the results from Lemma B.5 and Lemma B.6, we can directly compute the inner product:

$$\begin{aligned}
&\langle \mathbf{d}_p, \mathbf{d}_s \rangle = \left\langle \check{\mathbf{w}} - \left[\hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}\right], \lambda_{u_n} \operatorname{vec}(u_n q_r^\top)^{\otimes l} + \lambda_{v_r} \operatorname{vec}(v_r q_r^\top)^{\otimes l} \right\rangle \\
&= \lambda_{u_n} \left\langle \check{\mathbf{w}} - \left[\hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}\right], \operatorname{vec}(u_n q_r^\top)^{\otimes l} \right\rangle + \lambda_{v_r} \left\langle \check{\mathbf{w}} - \left[\hat{\mathbf{w}} + \eta \operatorname{vec}(u_n q_r^\top)^{\otimes l}\right], \operatorname{vec}(v_r q_r^\top)^{\otimes l} \right\rangle \\
&= \lambda_{u_n} \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l} \sigma_r \sqrt{\alpha} u_{n,j_\iota} q_{r,p_\iota} u_{n,j_\iota} q_{r,p_\iota} - \eta \prod_{\iota=1}^{l} u_{n,j_\iota} q_{r,p_\iota} u_{n,j_\iota} q_{r,p_\iota} \\
&+ \lambda_{v_r} \sum_{j_1=1,\cdots,j_l=1}^{n,\cdots,n} \sum_{p_1=1,\cdots,p_l=1}^{r,\cdots,r} \prod_{\iota=1}^{l} \sigma_r \sqrt{\beta} v_{r,j_\iota} q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota} - \prod_{\iota=1}^{l} \sigma_r v_{r,j_\iota} q_{r,p_\iota} v_{r,j_\iota} q_{r,p_\iota} \\
&= \lambda_{u_n} (\sigma_r^l \alpha^{\frac{l}{2}} - \eta) \|u_n\|_2^{2l} \|q_r\|_2^{2l} + \lambda_{v_r} \sigma_r^l (\beta^{\frac{l}{2}} - 1) \|v_r\|_2^{2l} \|q_r\|_2^{2l} \\
&= \lambda_{u_n} (\sigma_r^l \alpha^{\frac{l}{2}} - \eta) + \lambda_{v_r} \sigma_r^l (\beta^{\frac{l}{2}} - 1).
\end{aligned} \tag{89}$$

Finally, we establish the upper bound for the spectral norm of $\mathbf{d}_p$:

$$\|\mathbf{d}_p\|_S = |\langle \mathbf{d}_p, \mathbf{d}_s \rangle| \leq |\lambda_{u_n}| \cdot \left|\sigma_r^l \alpha^{\frac{l}{2}} - \eta\right| + |\lambda_{v_r}| \cdot \sigma_r^l \cdot \left|\beta^{\frac{l}{2}} - 1\right|. \tag{90}$$

As $\check{\mathbf{w}}$ aligns with the projection directions $\operatorname{vec}(u_n q_r^\top)^{\otimes l}$ and $\operatorname{vec}(v_r q_r^\top)^{\otimes l}$, we have $\sigma_r^l \alpha^{\frac{l}{2}} \to \eta$ and $\beta^{\frac{l}{2}} \to 1$. Consequently, we obtain:

$$\|\mathbf{d}_p\|_S \longrightarrow 0. \tag{91}$$

This completes the proof. □

This guarantees that, in terms of the spectral norm, the rank-1 tensor solution $\check{\mathbf{w}}$ approximates the desired solution $\mathbf{w}_{\text{escape}}$. This final step is analogous to perfectly sealing the edges of a dumpling, ensuring that the wrapper is neatly and tightly secured.

B.4 Proof of Theorem 4.4

This subsection is devoted to proving one of the most important theorems in our work: Theorem 4.4.

Since the validity of Theorem 4.3 depends on the condition $\beta \to 1$, we focus here on deriving the range of α that guarantees $\check{X}$ can escape the local minimum $\hat{X}$ in the matrix space. To accomplish this, we perform a Taylor expansion of Equation (3) around $\hat{X}$ and leverage the Restricted Strong Smoothness (RSS) property to analyze the solution.

In the first and second subsubsections, we compute the first-order and second-order terms of the Taylor expansion around $\hat{X}$. In the third subsubsection, these results are used to identify the range of α that satisfies the escape condition.

B.4.1 First-Order Taylor Expansion Term at $\hat{X}$

Lemma B.7. *The first-order Taylor expansion term of $f(XX^\top)$ at $\hat{X}\hat{X}^\top$ satisfies:*

$$\left\langle \nabla f(\hat{X}\hat{X}^\top), \check{X}\check{X}^\top - \hat{X}\hat{X}^\top \right\rangle = \lambda_n \sigma_r^2 \alpha, \tag{92}$$

where λ_n is the smallest eigenvalue of $\nabla f(\hat{X}\hat{X}^\top)$, σ_r represents the smallest singular of $\hat{X}$.

Proof. We start by expanding $\langle \nabla f(\hat{X}\hat{X}^\top), \check{X}\check{X}^\top - \hat{X}\hat{X}^\top \rangle$ as follows:

$$\begin{aligned}
&= \left\langle \nabla f(\hat{X}\hat{X}^\top), \sum_{\phi=1}^{r-1} \sigma_\phi^2 v_\phi v_\phi^\top + \sigma_r^2(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)^\top - \sum_{\phi=1}^{r} \sigma_\phi^2 v_\phi v_\phi^\top \right\rangle \\
&= \left\langle \nabla f(\hat{X}\hat{X}^\top), \sigma_r^2(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)^\top - \sigma_r^2 v_r v_r^\top \right\rangle \\
&= \left\langle \nabla f(\hat{X}\hat{X}^\top), \sigma_r^2 \left(\sqrt{\alpha}u_n u_n^\top \sqrt{\alpha} + \sqrt{\alpha}u_n v_r^\top \sqrt{\beta} + \sqrt{\beta}v_r u_n^\top \sqrt{\alpha} + \sqrt{\beta}v_r v_r^\top \sqrt{\beta} - v_r v_r^\top\right)\right\rangle \\
&= \left\langle \nabla f(\hat{X}\hat{X}^\top), \sigma_r^2 \left(\alpha u_n u_n^\top + \sqrt{\alpha\beta}u_n v_r^\top + \sqrt{\beta\alpha}v_r u_n^\top + (\beta - 1)\, v_r v_r^\top\right)\right\rangle \\
&= \left\langle \nabla f\left(\hat{X}\hat{X}^\top\right), \sigma_r^2 \alpha u_n u_n^\top \right\rangle
\end{aligned} \tag{93}$$

Next, we expand the quadratic term $(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)^\top$:

$$\begin{aligned}
&= \left\langle \nabla f(\hat{X}\hat{X}^\top), \sigma_r^2 \left[(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)(\sqrt{\alpha}u_n + \sqrt{\beta}v_r)^\top - v_r v_r^\top\right]\right\rangle \\
&= \left\langle \nabla f(\hat{X}\hat{X}^\top), \sigma_r^2 \left(\alpha u_n u_n^\top + \sqrt{\alpha\beta}u_n v_r^\top + \sqrt{\beta\alpha}v_r u_n^\top + (\beta - 1)v_r v_r^\top\right)\right\rangle.
\end{aligned} \tag{94}$$

The terms involving $\nabla f(\hat{X}\hat{X}^\top)v_r$ and $\nabla f(\hat{X}\hat{X}^\top)^\top v_r$ vanish due to the orthogonality condition $\nabla f(\hat{X}\hat{X}^\top)v_\phi = 0$ for all v_ϕ ($\phi = 1, \ldots, r$) and the fact that $\nabla f(\hat{X}\hat{X}^\top)$ is a symmetric matrix. Consequently, the only remaining non-zero term is:

$$\begin{aligned}
\left\langle \nabla f(\hat{X}\hat{X}^\top), \check{X}\check{X}^\top - \hat{X}\hat{X}^\top \right\rangle &= \left\langle \nabla f(\hat{X}\hat{X}^\top), \sigma_r^2 \alpha u_n u_n^\top \right\rangle = \sigma_r^2 \alpha \operatorname{Tr}\left(\nabla f(\hat{X}\hat{X}^\top)^\top u_n u_n^\top\right) \\
&= \sigma_r^2 \alpha \operatorname{Tr}\left(u_n^\top \nabla f(\hat{X}\hat{X}^\top)^\top u_n\right) = \sigma_r^2 \alpha \operatorname{Tr}\left(u_n^\top \nabla f(\hat{X}\hat{X}^\top) u_n\right) \\
&= \sigma_r^2 \alpha u_n^\top \nabla f(\hat{X}\hat{X}^\top) u_n = \lambda_n \sigma_r^2 \alpha
\end{aligned} \tag{95}$$

This completes the proof. ☐

B.4.2 Second-Order Taylor Expansion Term at $\hat{X}$

Lemma B.8. *The second-order Taylor expansion term of $f(XX^\top)$ at $\hat{X}\hat{X}^\top$ satisfies:*

$$\left\|\check{X}\check{X}^\top - \hat{X}\hat{X}^\top\right\|_F^2 = \sigma_r^4 \left[\alpha^2 + 2\alpha\beta + (\beta - 1)^2\right], \tag{96}$$

where σ_r represents the smallest singular value of $\hat{X}$.

Proof. The Frobenius norm squared can be expressed as:

$$\left\|\check{X}\check{X}^\top - \hat{X}\hat{X}^\top\right\|_F^2 = \operatorname{Tr}\left[(\check{X}\check{X}^\top - \hat{X}\hat{X}^\top)(\check{X}\check{X}^\top - \hat{X}\hat{X}^\top)^\top\right] \tag{97}$$

Substituting the expression of $\check{X}\check{X}^\top - \hat{X}\hat{X}^\top$, we have:

$$= \sigma_r^4 \operatorname{Tr}\left[\left(\alpha u_n u_n^\top + \sqrt{\alpha\beta}u_n v_r^\top + \sqrt{\beta\alpha}v_r u_n^\top + (\beta - 1)v_r v_r^\top\right)\left(\alpha u_n u_n^\top + \sqrt{\alpha\beta}u_n v_r^\top + \sqrt{\beta\alpha}v_r u_n^\top + (\beta - 1)v_r v_r^\top\right)\right]$$

Expanding the terms results in:

$$\begin{aligned}
&= \sigma_r^4 \operatorname{Tr}\left(\alpha u_n u_n^\top \alpha u_n u_n^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\alpha u_n u_n^\top \sqrt{\alpha\beta}u_n v_r^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\alpha u_n u_n^\top \sqrt{\beta\alpha}v_r u_n^\top\right) \\
&+ \sigma_r^4 \operatorname{Tr}\left(\alpha u_n u_n^\top (\beta - 1)v_r v_r^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\sqrt{\alpha\beta}u_n v_r^\top \alpha u_n u_n^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\sqrt{\alpha\beta}u_n v_r^\top \sqrt{\alpha\beta}u_n v_r^\top\right) \\
&+ \sigma_r^4 \operatorname{Tr}\left(\sqrt{\alpha\beta}u_n v_r^\top \sqrt{\beta\alpha}v_r u_n^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\sqrt{\alpha\beta}u_n v_r^\top (\beta - 1)v_r v_r^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\sqrt{\beta\alpha}v_r u_n^\top \alpha u_n u_n^\top\right) \\
&+ \sigma_r^4 \operatorname{Tr}\left(\sqrt{\beta\alpha}v_r u_n^\top \sqrt{\alpha\beta}u_n v_r^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\sqrt{\beta\alpha}v_r u_n^\top \sqrt{\beta\alpha}v_r u_n^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\sqrt{\beta\alpha}v_r u_n^\top (\beta - 1)v_r v_r^\top\right) \\
&+ \sigma_r^4 \operatorname{Tr}\left((\beta - 1)v_r v_r^\top \alpha u_n u_n^\top\right) + \sigma_r^4 \operatorname{Tr}\left((\beta - 1)v_r v_r^\top \sqrt{\alpha\beta}u_n v_r^\top\right) + \sigma_r^4 \operatorname{Tr}\left((\beta - 1)v_r v_r^\top \sqrt{\beta\alpha}v_r u_n^\top\right) \\
&+ \sigma_r^4 \operatorname{Tr}\left((\beta - 1)v_r v_r^\top (\beta - 1)v_r v_r^\top\right)
\end{aligned} \tag{98}$$

Using the orthogonality condition $u_n \perp v_r$, the cross terms involving $u_n^\top v_r$, $v_r^\top u_n$, $u_n v_r^\top$, and $v_r u_n^\top$ vanish. Furthermore, since $u_n^\top u_n = 1$ and $v_r^\top v_r = 1$, the remaining terms simplify to the following:

$$\begin{aligned}
&= \operatorname{Tr}\left[(\check{X}\check{X}^\top - \hat{X}\hat{X}^\top)(\check{X}\check{X}^\top - \hat{X}\hat{X}^\top)^\top\right] \\
&= \sigma_r^4 \operatorname{Tr}\left(\alpha u_n u_n^\top \alpha u_n u_n^\top\right) + \sigma_r^4 \operatorname{Tr}\left(\sqrt{\alpha\beta} u_n v_r^\top \sqrt{\beta\alpha} v_r u_n^\top\right) \\
&+ \sigma_r^4 \operatorname{Tr}\left(\sqrt{\beta\alpha} v_r u_n^\top \sqrt{\alpha\beta} u_n v_r^\top\right) + \sigma_r^4 \operatorname{Tr}\left((\beta-1) v_r v_r^\top (\beta-1) v_r v_r^\top\right) \\
&= \sigma_r^4\left[\alpha^2 + \alpha\beta + \beta\alpha + (\beta-1)^2\right] = \sigma_r^4\left[\alpha^2 + 2\alpha\beta + (\beta-1)^2\right] \qquad (99)
\end{aligned}$$

This completes the proof. □

B.4.3 Range of α for Escaping the Local Minimum $\hat{X}$

In this subsection, we formally present the proof of Theorem 4.4.

Proof. Let $\hat{M} = \hat{X}\hat{X}^\top$ and $\check{M} = \check{X}\check{X}^\top$. By performing a Taylor expansion of $f(M)$ around $\hat{M}$, we have:

$$f(\check{M}) = f(\hat{M}) + \left\langle \nabla f(\hat{M}), \check{M} - \hat{M} \right\rangle + \frac{1}{2}\left[\nabla^2 f(\tilde{M})\right]\left\langle \check{M} - \hat{M}, \check{M} - \hat{M} \right\rangle \qquad (100)$$

where $\tilde{M}$ is a convex combination of $\check{M}$ and $\hat{M}$.

Using the Restricted Smoothness (RSS) property, we further bound the second-order term:

$$\begin{aligned}
f(\check{M}) &< f(\hat{M}) + \left\langle \nabla f(\hat{M}), \check{M} - \hat{M} \right\rangle + \frac{L_s}{2}\left\|\check{M} - \hat{M}\right\|_F^2 \\
&= f(\hat{M}) + \lambda_n \sigma_r^2 \alpha + \frac{L_s}{2}\sigma_r^4\left[\alpha^2 + 2\alpha\beta + (\beta-1)^2\right] \qquad (101)
\end{aligned}$$

where the first-order term is derived from Lemma B.7, and the second-order term is from Lemma B.8.

For $f(\check{M})$ to be smaller than $f(\hat{M})$, it suffices to ensure:

$$\begin{aligned}
&\lambda_n \sigma_r^2 \alpha + \frac{L_s}{2}\sigma_r^4\left[\alpha^2 + 2\alpha\beta + (\beta-1)^2\right] < 0 \\
&\Leftrightarrow \frac{L_s}{2}\sigma_r^2\left[\alpha^2 + 2\alpha\beta + (\beta-1)^2\right] < -\lambda_n \alpha \\
&\Leftrightarrow \frac{L_s}{2}\sigma_r^2\left[\alpha + 2\beta + \frac{(\beta-1)^2}{\alpha}\right] < -\lambda_n \qquad (102)
\end{aligned}$$

Based on the approximation condition in Theorem 4.3, assume $\beta = 1$. Substituting $\beta = 1$ into the inequality gives:

$$\frac{L_s}{2}\sigma_r^2(\alpha + 2) < -\lambda_n \Leftrightarrow \alpha < \frac{-2\lambda_n}{L_s \sigma_r^2} - 2 \qquad (103)$$

This completes the proof. □

Under this condition, $f(\check{X}\check{X}^\top)$ is guaranteed to be smaller than $f(\hat{X}\hat{X}^\top)$, which implies that the local minimum at $\hat{M} = \hat{X}\hat{X}^\top$ is escaped in the matrix space.

C Experimental Details and Validation

C.1 Experimental Environment

All experiments were conducted on a personal laptop running Ubuntu as the operating system. The computations were accelerated using a GeForce GTX 1660 Ti GPU with CUDA version 11.1 and cuDNN version 8.0.5.

For efficient matrix and tensor computations, the experiments utilized JAX (version 0.3.8) [37], which leverages GPU acceleration through CUDA and cuDNN. Additionally, Optax (version 0.1.4) was employed for optimization routines, and Chex (version 0.1.6) was used for testing and debugging. Both Optax and Chex are part of the DeepMind JAX Ecosystem [38].

This setup provided a robust and efficient computational environment, enabling precise and high-performance experimentation.

C.2 Numerical Validation of Theoretical Bounds

In this subsection, we present the specific numerical results from the perturbed matrix completion experiments discussed in the main text, focusing on the case where $n = 11$ under the application of LEAC-Simulated.

Specifically, we consider $M^* = ZZ^T$, where:

$$Z = [1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]^\top \Rightarrow \mathrm{Tr}(M^*) = 6.0 \tag{104}$$

The sensing matrix was constructed based on the formulation in Equation (17). The computed Restricted Strong Smoothness (RSS) constant and Restricted Strong Convexity (RSC) constant are:

$$L_s = 0.9987, \qquad \alpha_s = 0.2638 \tag{105}$$

The algorithm encountered a local minimum at approximately the 50th iteration, at which point the following values were observed:

$$\sigma_r = 1.3923e-7, \qquad \lambda_n = -1.00025, \tag{106}$$

$$v_r = [0.6168, 0.1730, -0.0742, 0.0171, 0.0160, -0.4711, -0.4043, 0.2654, -0.2097, -0.2782, 0.0806]^\top \tag{107}$$

$$q_r = [1.0], \tag{108}$$

$$u_n = [0.4083, 0.0, 0.4085, 0.0, 0.4081, 0.0, 0.4083, 0.0, 0.4082, 0.0, 0.4082]^\top \tag{109}$$

$$\|\hat{X}\hat{X}^\top - M^*\|_F^2 = 36.0. \tag{110}$$

At this point, the inequality in Equation (10) holds, which is critical for our algorithm:

$$1.00025 = -\lambda_n \geq 0.7915 = \frac{\alpha_s \|\hat{X}\hat{X}^\top - M^*\|_F^2}{2\,\mathrm{Tr}(M^*)} \geq \frac{L_s \sigma_r^2}{2} = 9.6793e-15 \tag{111}$$

The upper bound of α for escaping the local minimum is computed as follows:

$$\Upsilon = \frac{-2\lambda_n}{L_s \sigma_r^2} - 2 = 103339150000000 \Rightarrow \eta = \sigma_r^l \alpha^{\frac{l}{2}} < \sigma_r^l \Upsilon^{\frac{l}{2}} = 2.8352 \tag{112}$$

This result demonstrates that the step size required to escape in the matrix space (measured by $\sqrt{\alpha}$) is significantly larger than the step size required to escape in the tensor space (measured by η).

Next, we compute $\check{X}$ based on Equation (14) and derive its corresponding tensor $\check{\mathbf{w}}$ in the lifted tensor space. The distances between $\check{\mathbf{w}}$ and $\hat{\mathbf{w}}$, as well as $\check{\mathbf{w}}$ and $\mathbf{w}_{\text{escape}}$, are given as:

$$\|\check{\mathbf{w}} - \hat{\mathbf{w}}\| = 2.8352 \tag{113}$$

$$\|\check{\mathbf{w}} - \mathbf{w}_{\text{escape}}\| = 5.6335e-7 \tag{114}$$

These results confirm that the tensor $\check{\mathbf{w}}$ is indeed a rank-1 tensor that closely approximates $\mathbf{w}_{\text{escape}}$.

Finally, we compute the objective function values at $\check{X}$ and $\hat{X}$ using Equation (3):

$$1.3319 = f(\check{X}\check{X}^\top) < f(\hat{X}\hat{X}^\top) = 3.0015 \tag{115}$$

This demonstrates that our LEAC-Simulated mechanism, guided by the oracle from the tensor space, deterministically escapes the local minimum $\hat{X}$ in a single step to reach the point $\check{X}$.